Hash Proofs Systems and Applications to Implicit Cryptography

Thèse d'habilitation

présentée et soutenue publiquement le 12 juin 2019 par

Olivier Blazy

pour l'obtention du

Diplôme d'Habilitation à Diriger des Recherches
de l'Université de Limoges

Devant le jury composé de :

Correspondant HDR :	Philippe Gaborit	(Université de Limoges)
Rapporteurs :	Dennis Hofheinz	(Karlsruhe Institute of Technology)
	Fabien Laguillaumie	(Université Claude Bernard Lyon 1)
	Ayoub Otmani	(Université de Rouen)
Examinateurs :	Tibor Jager	(Université de Paderborn)
	Duong-Hieu Phan	(Université de Limoges)
	David Pointcheval	(CNRS / École Normale Supérieure)
	Damien Vergnaud	(Sorbonne Université / Institut Universitaire de France)

Travaux effectués au Laboratoire XLim de l'Université de Limoges

Acknowledgments

A lot of things could be said today, there have been so many changes in the 2626 days since my PhD defense. Some of them being much more joyful than others, some being much more forced... This habilitation is the occasion to do a status report on the scientific side of those last years, force myself to take some time to breath, and decide on what to focus next. Let's now start the proper acknowledgments, I am hoping not to forget anyone, but please don't see any malice in any omission.

Members of the jury always appear first in this section. I'd like to thank Philippe Gaborit for accepting to coordinate the administrative side of this habilitation process. Thank you for the warm welcome in Limoges and for making this place a nice environment to work in. I also want to thank Dennis Hofheinz, Fabien Laguillaumie and Ayoub Otmani for accepting to be *Rapporteurs* of this thesis, and also the rest of the jury Tibor Jager, Duong-Hieu Phan, David Pointcheval, and Damien Vergnaud, thank you for making this possible, and for sacrificing part of your time in this busy period.

One cannot do research alone, and I would also like to thank every colleagues I had the occasion to chat, work, exchange with. The list would be quite long, and I am afraid to miss someone, but a big thank you to the people from the Cryptis team here in Limoges, to those from the FoC group in Bochum, to the current and former member of the Cascade team from ENS, and to people in the Crypto group from Louvain-la-Neuve where I did my first crypto steps. I hope we will continue to have numerous discussions, and continue to make the crypto community an enjoyable work environment. I also want to thank the various members of the administrative staff for their everyday support.

Not only is the crypto community a friendly environment but sometimes we also produce papers. For this I would like to thank all my co-authors, some of which I never had the occasion to meet. In those that were not explicitly mentioned earlier, I have to thank Céline Chevalier, thanks for all those discussions, supporting my weird ideas, maybe some day we will manage to write a paper where none of us say at some point "you are not clear". There are so many other co-authors I'd like to thank, Pascal Lafourcade for his energy, and his ideas of fun constructions, thank you also to Paul, Laura and Neals, for being *my* first PhD Students.

Professionally I am torn between 50% of research, 50% of teaching, and 50% of administrative work. I really want to thank members of the Cryptis Team / Computer Science department for making this a pleasant experience. Thank you Damien, Emmanuel for helping make the best of a bad situation.

After a while, the line gets blurry between colleagues, friends; and a very serious discussion about cryptographic protocols can derive into an exchange of weird narwhal pictures. I would have never thought there was a link between obfuscation and Bollywood movies, and the omnipresence of acronyms make confusing discussions about TOR, and not TOR... A very special thank you to Amandine, David, Isis, and Saqib for those discussions, and I hope we will have many others.

Playing an important part in keeping my sanity are also the non crypto friends (yes, they exist!). Thank you for being present when I need it, and thank you for all the wonderful adventures. And sorry to have bothered you way more than what is reasonable with my manuscript and slides.

And finally, I of course want to thank my family. Thank you for always pushing me, always believing in me. Thank you for being here when I needed it, and for supporting my crazy hours during deadlines throughout the years.

Je sers la science et c'est ma joie.

Contents

Chapter 1

Introduction

Contents

This thesis presents research done by the author (and several co-authors) since his doctoral thesis. Due to space constraints, we will focus on one principal research theme, only works related to the design of protocols in public-key cryptography with implicit authentication are presented. Our research works in other domains of cryptography, such as code-based cryptography [ABCG17, AMBD$^+$18] with the various NIST submissions, constructions with tight security [BKP14, BKKP15]) are not presented. A complete list of publications is available on Section 1.5, page 4.

Everyday, we see an increase in the importance of security and privacy. Cryptography must help the modern world protect individuals, preserve the privacy of their life, and all interactions therein.

Nearly 30 years ago, [GMR89] proposed the concept of Zero-Knowledge proofs to revolutionize the way we handle secret informations. The idea was to prove that a statement was fulfilled without revealing anything else. Such proofs have lead to a wonderful array of applications like [BW06, Gro07, FPV09, BFI$^+$10, JR13].

However, we now know that metadata is often enough to learn enough information to learn someone's secret. For example, even if we don't know what is said, we can paint a pretty clear picture of what is happening when someone gets a call from an STD specialist, calls some former contacts, and then a suicide hotline.

In this context, knowing that a statement exists and is fulfilled is already too much, and such problem lead to the study of implicit proofs. Some existing primitives were already designed in the context of implicit cryptography. In case of two party protocols, we have shown that they can be classified into two main categories.

1.1 Symmetric Primitives

In the category of symmetric primitives, we suppose two users interact in order to generate a shared high entropy secret key. As often with public key cryptography, this shared key will later be used with a symmetric primitive.

One of the most widely studied problems in this area is called **Password Authenticated Key Exchange** as introduced by Bellovin and Merritt [BM92], where two users possess a password (more formally a low entropy secret), and they interact so that they obtain a shared (high entropy) key if and only if their password match. Of course, we expect that they learn nothing of the other's password otherwise.

There exists several variants of this primitive in the literature, Secret Handshakes has been introduced in 2003 by Balfanz *et al.* [BDS$^+$03] (see also [JL09b, AKB07]) echoing to the Masonic handshake that would supposedly allow members to identify with each other. It allows two members of the same group to identify each other secretly, in the sense that each party reveals his affiliation to the other only if

they are members of the same group. At the end of the protocol, the parties can set up an ephemeral session key for securing further communication between them and an outsider is unable to determine if the handshake succeeded. In case of failure, the players do not learn any information about the other party's affiliation. Here users would possess a signature by an authority and would manage to obtain a shared key, if and only if both signatures were made by the same authority. Another variant, proposed in [CCGS10] would now consider secret Credentials, and this CAKE (**Credential Authenticated Key Exchange**) would succeed if and only if the secret credentials match.

In all those cases, the users are expected to possess secrets (words), and are expected to obtain a shared secret if and only if, those words belong to a specific language, which lead us to supersede those notions with the idea of LAKE (**Language Based Authenticated Key Exchange**) [BBC+13b].

1.2 Asymmetric Primitives

On the other hand, there are cases where a user is interacting with a server, and we do not expect both of the participant to behave the same way. For example, when secretly retrieving data, the user expect the server not to learn which line was requested, but the server might only expect the user not to be able to retrieve more than one line at once.

This is a primitive called **Oblivious Transfer**, which was originally introduced by Rabin [Rab81]. Oblivious Transfer is at the heart of various protocols from Yao's protocol [Yao86], to Oblivious RAM [WHC+14,GOS18] or even most garbled circuit [BHR12]. It comes to no surprise, that Oblivious Transfer are everywhere since Kilian [Kil88] showed that every multi-party computation scheme can be achieved from an oblivious transfer.

Other notions have echoed to Oblivious Transfer since, **Private Information Retrieval** [CGKS95] is a relaxed version of Oblivious Transfer, where the server still need to be oblivious to the retrieved data but the user is allowed to learn more. With the increasing need for user privacy, these schemes are quite useful in practice and may be used for accessing records for email repositories, collection of webpages, music... Even though protecting the privacy of the user, it is equally important that the user should not learn more information than he is allowed to. This is known as database privacy and the corresponding protocol is called a Symmetrically Private Information Retrieval (**SPIR**), which could be employed in practice, for medical data or biometric information

One can also perform a conditional Oblivious Transfer, expecting the user to possess some information to be able to access some information, a special case would be for signature through **Oblivious Signature-Based Envelope**. It was introduced by Li, Du and Boneh in [LDB03]. **OSBE** schemes consider the case where Alice (the receiver) is a member of an organization and possesses a certificate produced by an authority attesting she actually belongs to this organization. Bob (the sender) wants to send a private message P to members of this organization. However due to the sensitive nature of the organization, Alice does not want to give Bob neither her certificate nor a proof she belongs to the organization. OSBE lets Bob send an obfuscated version of this message P to Alice, in such a way that she will be able to find P if and only if she is in the required organization. In the process, Bob cannot decide whether Alice does really belong to the organization.

Once again, the access to an information is gated behind the possession of a word, belonging to a specific language, which we generalized into **Oblivious Language-Based Envelope** [BCG16] encompassing most examples of conditional disclosure schemes: Access Controlled Oblivious Transfer [CDN09, CDNZ11], Priced Oblivious Transfer [AIR01, RKP09]) and Conditional Oblivious Transfer [DOR99], and Conditional Disclosure of Secrets (see for instance [AIR01, GIKM98, BGN05, LL07, Wee14, IW14, Att14, GKW15]).

1.3 Contributions

First, I will focus (chronologically) on five of the more meaningful publications in my recent years, and then I will briefly sketch other relevant results that will not necessarily be detailed in this manuscript.

New Techniques for SPHFs and Efficient One-round PAKE Protocols [BBC+13b] At Crypto'13, we managed to classify existing Smooth Projective Hash Function into different families, we showed that one of those families (named KV as it first was used in [KV11]) are of particular interest as it allowed to build round-optimal protocol. We managed to propose the first so-called KV-SPHF on Cramer Shoup Encryption, leading to the most efficient PAKE scheme in the BPR model. It was an open problem at the time, and the solution used by [KV11] lacked efficiency, as it consisted in building an SPHF over

ElGamal, and then using a Simulation-Sound Zero-Knowledge proof to transform it into a CCA scheme, it also required living in a bilinear group as it used pairings.

An interesting side-results in this paper are Trapdoor-SPHF, that allows some simulation in the proof at the cost of having *only* a computational Smoothness.

(Hierarchical) Identity-Based Encryption from Affine Message Authentication [BKP14] At Crypto'14, we proposed a generic transformation from any affine message authentication code (MAC) to an identity-based encryption (IBE) scheme over pairing groups of prime order. If the MAC satisfies a security notion related to unforgeability against chosen-message attacks and, for example, the k-Linear assumption holds, then the resulting IBE scheme is adaptively secure. Our security reduction is tightness preserving, i.e., if the MAC has a tight security reduction so has the IBE scheme. Furthermore, the transformation also extends to hierarchical identity-based encryption (HIBE). We also showed how to construct affine MACs with a tight security reduction to standard assumptions.

While this result was not directly related to Implicit Authentication, we will see in Section 7.3, page 47, that it can be used to design Adaptive Oblivious Transfer. Interestingly, this is one of the case where having a all-powerful authority generating each user secret key is not a liability, as the authority is also going to be the performing the encryption.

Adaptive Oblivious Transfer and Generalization [BCG16] Oblivious Transfer are famous in theoretical cryptography as one of the main building block for many Multi-Party Computation scheme. However they fail to find many practical applications due their inefficiency. In Asiacrypt'16, we proposed an Adaptive Oblivious Transfer, UC secure with semi-adaptive corruption. This scheme, still has the huge overhead inherent to UC-secure Oblivious Transfer, however if a user queries adaptively more than one line, the server no longer has to resend everything allowing much faster communication.

While doing so, we also proposed a generalization of Oblivious Transfer protocols in the same vein as the LAKE we introduced in PKC'13. This allowed us to highlight core-building blocks in SPHF-based Oblivious Transfer constructions (and Oblivious Signature-Based Envelope, Credentials-based schemes, ...). We gave an ideal functionality, and a generic construction fulfilling it which we then use to instantiate efficient version of various schemes.

Structure-Preserving Smooth Projective Hashing [BC16] Still at Asiacrypt'16, we proposed in another paper, a new concept we named Structure-Preserving Smooth Projective Hash Function. While it was never explicitly required, classical SPHF built on elliptic curve were using a scalar as witness. This was really simple to handle, however it destroys any hope in having some simulation freedom in the witness. Our new notion, in the same vein as Structure-Preserving signature, requires projections keys, words, and also witnesses to be group elements.

This allows us to use the UC commitment from [FLM11] with an SP-SPHF. We show that when doing that, we can instantiate compact PAKE and Oblivious Transfer Schemes. Contrarily to pre-existing schemes, these schemes only had constant size flows (minus the database overhead for OT) and no longer the logarithmic size due to the bit per bit commitment. It should be noted that during the cycle of submissions [JR15] appeared and proposed a more efficient PAKE based on a QA-NIZK approach.

Efficient Encryption From Random Quasi-Cyclic Codes [AMBD+18] In IEEE Transactions of Information Theory of May 2018, we proposed a framework for constructing efficient code-based encryption schemes. It had the particularity of not hiding any structure in their public matrix. The framework is in the spirit of the schemes first proposed by Alekhnovich in 2003 [Ale03] and based on the difficulty of decoding random linear codes from random errors of low weight. We somewhat depart from Alekhnovich's approach and propose an encryption scheme based on the difficulty of decoding random quasi-cyclic codes.

We proposed two new cryptosystems within our framework: the hamming quasi-cyclic cryptosystem (HQC), based on the hamming metric, and the rank quasi-cyclic cryptosystem (RQC), based on the rank metric. We give a security proof, which reduces the indistinguishability under chosen plaintext attack security of our systems to a decision version of the well-known problem of decoding random families of quasi-cyclic codes for the hamming and rank metric.

This scheme was a part of several works destined to be submitted to the NIST Post-Quantum competition. This one is particular as it has a structure clean enough for us to be able to build the very first code-based SPHF on it. Interestingly due to the difference in Metric, one can manage to avoid the

pitfall of having to consider difference languages for the correctness and the smoothness that is present on Lattice-based constructions.

1.4 Summary of other results

While I have several results that can not be efficiently categorized in a family, one can still distinguish three main topic of interest in my recent publications.

UC protocols We have had the occasion of working and proving several protocols in the UC framework. We revisited (and improved) Lindell's Commitment in [BCPV13] managing to reduce the number of required rounds and revisiting the security.

We also proposed various ameliorations / generalizations around PAKE and Oblivious Transfer in [BBC$^+$13a, ABB$^+$13, BC15, BCG17]. The first paper presented a generalization encompassing most (if not all) AKE protocols, we proposed a UC functionality, and a generic construction. The 2nd and then the 3rd paper proposed generic ways to build Oblivious Transfer first for pairing-based cryptography, and then for several hypotheses as long as the basis tool were existing. The last one, proposed a transformation from PAKE to Oblivious Transfer and proposed the most efficient scheme to date.

Post-Quantum Cryptography When trying to consider other underlying hypotheses, a natural contender in the post-quantum world is to have a look at Lattice-based cryptography. While there was already a proposal from [KV09] several issues make it an unreasonable building blocks. In [BBDQ18], we proposed an SPHF for lattices that is efficient, and work directly on classical schemes.

Due to the general expertise, of my current team, we have also proposed several results around code-based cryptography [ABCG15, ABCG16, ABCG17, AMBD$^+$18, ABG$^+$19]. This was the occasion to formalize some results to start closing the gap in term of functionality between code-based and lattice-based crypto, and also we submitted several contenders (7) to NIST PQC competition, still in the race as of this writing. The last paper of the list is a proper code-based signature that emerged too late for the competition.

Tight Security We also have some results around Tight Security [BKP14, BKKP15], in both cases we design generic construction of tight primitive (IBE and Signatures) from simple building blocks. For IBE's, when applying we manage to circumvent an impossibility result by proving a randomizable signature with only 3 groups elements. The trick is behind the definition of *randomizable* our signature does not randomize in the whole set of valid signature, but in an indistinguishable subset. For the signatures, we provide tight signature with a tree-based approach on chameleon hashes. One of our construction leads to a tight signature in the standard model based on discrete logarithm. Applying a garbled circuit on our construction based on DDH, one can obtain the IBE from DDH from [DG17].

1.5 List of Publications

[ABB$^+$13] Michel Abdalla, Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, and David Pointcheval. SPHF-Friendly Non-Interactive Commitment Schemes. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology - Proceedings of ASIACRYPT '13*, volume 8269 of *Lecture Notes in Computer Science*, pages 214–234, Bangalore, India, December 2013. Springer.

[ABCG15] Quentin Alamélou, Olivier Blazy, Stéphane Cauchie, and Philippe Gaborit. A Code-Based Group Signature Scheme. In Jean-Pierre Tillich, Pascale Charpin, Nicolas Sendrier, editor, *The 9th International Workshop on Coding and Cryptography 2015 WCC2015*, Paris, France, April 2015.

[ABCG16] Quentin Alamélou, Olivier Blazy, Stéphane Cauchie, and Philippe Gaborit. A practical group signature scheme based on rank metric. In Sylvain Duquesne and Svetla Petkova-Nikova, editors, *Arithmetic of Finite Fields - 6th International Workshop, WAIFI 2016, Ghent, Belgium, July 13-15, 2016, Revised Selected Papers*, pages 258–275. Springer, July 2016.

[ABCG17] Quentin Alamélou, Olivier Blazy, Stéphane Cauchie, and Philippe Gaborit. A code-based group signature scheme. *Designs, Codes and Cryptography*, 82:1–25, 2017.

[ABD$^+$18] Carlos Aguilar Melchor, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, and Gilles Zémor. Efficient encryption from random quasi-cyclic codes. *IEEE Trans. Information Theory*, 64(5):3927–3943, 2018.

[ABG$^+$19] Nicolas Aragon, Olivier Blazy, Philippe Gaborit, Adrien Hauteville, and Gilles Zémor. Durandal: a rank metric based signature scheme. To appear in Eurocrypt 2019.

In Jean-Pierre Tillich, Pascale Charpin, Nicolas Sendrier, editor, *The 9th International Workshop on Coding and Cryptography 2015 WCC2015*, Paris, France, April 2015.

[BBB+19a] Olivier Blazy, Angèle Bossuat, Xavier Bultel, Pierre-Alain Fouque, Cristina Onete, and Elena Pagnin. SAID: Reshaping Signal into an Identity-Based Asynchronous Messaging Protocol with Authenticated Ratcheting. To appear in EuroS&P 2019.

[BBC+13a] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. Efficient UC-Secure Authenticated Key-Exchange for Algebraic Languages. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Conference on Practice and Theory in Public-Key Cryptography (PKC '13)*, volume 7778 of *Lecture Notes in Computer Science*, pages 272–291, Nara, Japan, March 2013. Springer.

[BBC+13b] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. New Techniques for SPHFs and Efficient One-Round PAKE Protocols. In Ran Canetti and Juan Garay, editors, *Advances in Cryptology - Proceedings of CRYPTO '13*, volume 8042 of *Lecture Notes in Computer Science*, pages 449–475, Santa Barbara, California, August 2013. Springer.

[BBDQ18] Fabrice Benhamouda, Olivier Blazy, Léo Ducas, and Willy Quach. Hash proof systems over lattices revisited. In *Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, Proceedings, Part II*, volume 10770, pages 644–674. Springer, March 2018.

[BBL16a] Olivier Blazy, Xavier Bultel, and Pascal Lafourcade. Anonymizable ring signature without pairing. In Frédéric Cuppens, Lingyu Wang, Nora Cuppens-Boulahia, Nadia Tawbi, and Joaquín García-Alfaro, editors, *Foundations and Practice of Security - 9th International Symposium, FPS 2016, Québec City, QC, Canada, October 24-25, 2016, Revised Selected Papers*, pages 214–222, Québec, Canada, 2016. Springer.

[BBL16b] Olivier Blazy, Xavier Bultel, and Pascal Lafourcade. Two secure anonymous proxy-based data storages. In Christian Callegari, Marten van Sinderen, Panagiotis G. Sarigiannidis, Pierangela Samarati, Enrique Cabello, Pascal Lorenz, and Mohammad S. Obaidat, editors, *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications (ICETE 2016) - Volume 4: SECRYPT, Lisbon, Portugal, July 26-28, 2016.*, pages 251–258. Springer, July 2016.

[BC15] Olivier Blazy and Céline Chevalier. Generic Construction of UC-Secure Oblivious Transfer. In Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis, editors, *Applied Cryptography and Network Security - 13th International Conference, ACNS 2015, New York, NY, USA, June 2-5, 2015, Revised Selected Papers*, volume 9092 of *Lecture Notes in Computer Science*, pages 65–86, New York, USA, June 2015. Springer.

[BC16] Olivier Blazy and Céline Chevalier. Structure-preserving smooth projective hashing. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*, pages 339–369, Hanoi, Vietnam, December 2016. Springer.

[BC18a] Olivier Blazy and Céline Chevalier. Non-interactive key exchange from identity-based encryption. *Proceedings of the 13th International Conference on Availability, Reliability and Security, Hamburg, Germany, August 27 - August 30, 2018*, August 2018.

[BC18b] Olivier Blazy and Céline Chevalier. Spreading alerts quietly: New insights from theory and practice. *Proceedings of the 13th International Conference on Availability, Reliability and Security, Hamburg, Germany, August 27 - August 30, 2018*, August 2018.

[BCB+17] Olivier Blazy, Emmanuel Conchon, Pierre-François Bonnefoi, Damien Sauveron, Raja Naeem Akram, Konstantinos Markantonakis, Keith Mayes, and Serge Chaumette. An efficient protocol for uas security. In *Integrated Communications, Navigation and Surveillance Conference (ICNS), 2017*. IEEE, 2017.

[BCF+11] Olivier Blazy, Sébastien Canard, Georg Fuchsbauer, Aline Gouget, Hervé Sibert, and Jacques Traoré. Achieving optimal anonymity in transferable e-cash with a judge. In Abderrahmane Nitaj and David Pointcheval, editors, *AFRICACRYPT 2011 - 4th International Conference on Cryptology in Africa*, volume 6737 of *Lecture Notes in Computer Science*, pages 206–223, Dakar, Senegal, June 2011. Springer.

[BCG16] Olivier Blazy, Céline Chevalier, and Paul Germouty. Adaptive oblivious transfer and generalization. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*, pages 217–247, Hanoi, Vietnam, December 2016. Springer.

[BCG17] Olivier Blazy, Céline Chevalier, and Paul Germouty. Almost optimal oblivious transfer from QA-NIZK. In Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi, editors, *Applied Cryptography and Network Security - 15th International Conference, ACNS 2017, Kanazawa, Japan, July 10-12, 2017, Proceedings*, volume 10355 of *Lecture Notes in Computer Science*, pages 579–598. Springer, 2017.

[BCGJ17] Olivier Blazy, Emmanuel Conchon, Paul Germouty, and Amandine Jambert. Efficient id-based designated verifier signature. In *Proceedings of the 12th International Conference on Availability, Reliability and Security, Reggio Calabria, Italy, August 29 - September 01, 2017*, pages 44:1–44:8. ACM, 2017.

[BCPV12] Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. Efficient UC-Secure Authenticated Key-Exchange for Algebraic Languages. Technical report, IACR ePrint Archive, May 2012.

[BCPV13] Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. Analysis and Improvement of Lindell's UC-Secure Commitment Schemes. In Rei Safavi-Naini and Michael E. Locasto, editors, *Conference on Applied Cryptography and Network Security (ACNS '13)*, volume 7954 of *Lecture Notes in Computer Science*, pages 534–551, Banff, Alberta, Canadan, June 2013. Springer.

[BCV15] Olivier Blazy, Céline Chevalier, and Damien Vergnaud. Non-Interactive Zero-Knowledge Proofs of Non-Membership. In K. Nyberg, editor, *Proceedings of CT-RSA*, volume 9048 of *Lecture Notes in Computer Science*, pages 145–164, San Francisco, California, April 2015. Springer.

[BCV16] Olivier Blazy, Céline Chevalier, and Damien Vergnaud. Mitigating server breaches in password-based authentication: Secure and efficient solutions. In Kazue Sako, editor, *Topics in Cryptology - CT-RSA 2016 - The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29 - March 4, 2016, Proceedings*, pages 3–18, San Francisco, USA, February 2016. Springer.

[BDSS16] Olivier Blazy, David Derler, Daniel Slamanig, and Raphael Spreitzer. Non-interactive plaintext (in-)equality proofs and group signatures with verifiable controllable linkability. In Kazue Sako, editor, *Topics in Cryptology - CT-RSA 2016 - The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29 - March 4, 2016, Proceedings*, pages 127–143, San Francisco, USA, February 2016. Springer.

[BFI+10] Olivier Blazy, Georg Fuchsbauer, Malika Izabachène, Amandine Jambert, Hervé Sibert, and Damien Vergnaud. Batch groth-sahai. In Jianying Zhou and Moti Yung, editors, *Conference on Applied Cryptography and Network Security (ACNS '10)*, volume 6123 of *Lecture Notes in Computer Science*, pages 218–235, Beiing, China, June 2010. Springer.

[BFPV11] Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Signatures on Randomizable Ciphertexts. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *Conference on Practice and Theory in Public-Key Cryptography (PKC '11)*, volume 6571 of *Lecture Notes in Computer Science*, pages 403–422, Taormina, Italy, March 2011. Springer.

[BFPV13] Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Short Blind Signatures. *Journal of Computer Security*, 21(5):627–661, November 2013.

[BGP19] Olivier Blazy, Paul Germouty, Duong-Hieu Phan. Downgradable Identity-Based Encryption and Applications. In M. Matsui, editor, *Proceedings of CT-RSA*, volume 11405 of *Lecture Notes in Computer Science*, pages 44–61, San Francisco, California, Mach 2019. Springer.

[BGSS17] Olivier Blazy, Philippe Gaborit, Julien Schrek, and Nicolas Sendrier. A code-based blind signature. In *2017 IEEE International Symposium on Information Theory, ISIT 2017, Aachen, Germany, June 25-30, 2017*, pages 2718–2722, 2017.

[BKKP15] Olivier Blazy, Saqib A. Kakvi, Eike Kiltz, and Jiaxin Pan. Tightly-Secure Signatures from Chameleon Hash Functions. In Jonathan Katz, editor, *Conference on Practice and Theory in Public-Key Cryptography (PKC '15)*, volume 9020 of *Lecture Notes in Computer Science*, pages 257–278, Gaithersburg, Maryland, USA, 2015. Springer.

[BKP14] Olivier Blazy, Eike Kiltz, and Jiaxin Pan. (Hierarchical) Identity-Based Encryption from Affine Message Authentication. In Ran Canetti and Juan Garay, editors, *Advances in Cryptology - Proceedings of CRYPTO '14*, volume 8616 of *Lecture Notes in Computer Science*, pages 408–426, Santa Barbara, California, August 2014. Springer.

[BP12] Olivier Blazy and David Pointcheval. Traceable Signature with Stepping Capabilities. In David Naccache, editor, *Cryptography and Security: From Theory to Applications*, volume 6805 of Lecture Notes in Computer Science, pages 108–131. Springer, January 2012. Cryptography and Security: From Theory to Applications - Essays Dedicated to Jean-Jacques Quisquater on the Occasion of

His 65th Birthday.

[BPV12a] Olivier Blazy, David Pointcheval, and Damien Vergnaud. Compact round-optimal partially-blind signatures. In Ivan Visconti and Roberto De Prisco, editors, *The 8th Conference on Security in Communication Networks (SCN '12)*, volume 7485 of *Lecture Notes in Computer Science*, pages 95–112, Amalfi, Italy, September 2012. Springer.

[BPV12b] Olivier Blazy, David Pointcheval, and Damien Vergnaud. Round-Optimal Privacy-Preserving Protocols with Smooth Projective Hash Functions. In Ronald Cramer, editor, *9th Theory of Cryptography Conference (TCC '12)*, volume 7194 of *Lecture Notes in Computer Science*, pages 94–111, Taormina, Italy, March 2012. Springer.

Chapter 2

Technical Introduction

Contents

2.1 Universal Composability

In part II, page 33, our main goal will be to provide protocols security in the universal composability framework. This framework was introduced in [Can01]. While this framework can be quite overwhelming, we aim to give a brief overview to have some common conventions.

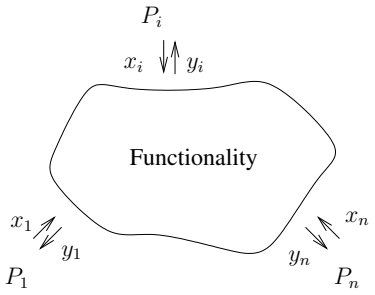
In the context of multi-party computation, one wants several users P_i with inputs x_i to be able to compute a specific function $f(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$ without learning anything except y_i, even when several players are colluding / cheating.

This approach was seen for example in Yao's Millionaires' problem [Yao82], where two millionaires want to know who is richer without revealing their respective wealth. So here, x_i is the wealth of the millionaire i, and f simply returns which one is richer (in this specific case $y_1 = y_i = y_n$). It should be noted, that there are several cases (Blind Signature, Oblivious Transfer, ...) where the outputs y_* are different for the various players involved.

Instead of following the classical approach which aims to list exhaustively all the expected properties, Canetti did something else and tried to define how a protocol should ideally work.

For that, he divided the world into two spaces, the real world, where the protocol is run with some possible attack, and the ideal world where everything would go smoothly. For a good protocol, it should be impossible to distinguish the real world from the ideal one.

In the ideal world there is an incorruptible entity named the ideal functionality, to which players can send their inputs privately, and then receive the corresponding output without any kind of communication between the players. This way the functionality can be set to be correct, without revealing anything except what is expected.



A protocol, in the real world with an adversary, should create an execution similar to the one obtained by the ideal functionality. This means that the communication between the players should not give more

information than the functionality description, and its output. In this case the protocol runs not really against the adversary but against the environment who picks the inputs given to the players, and obtains the outputs. After the interaction the environment should output a bit saying whether he is in the real world.

The main constraint is that the adversary is now free to interact with the environment whenever he wants which prevents the simulator from rewinding when needed. The adversary has access to the communication between the players but not their inputs/outputs, while the environment has only access to the inputs/outputs.

The goal of the framework [Can01] is to ensure that UC-secure protocols will continue to behave in the ideal way even if executed in a concurrent way in arbitrary environments. It is a simulation-based model, relying on the indistinguishability between the real world and the ideal world. In the ideal world, the security is provided by an ideal functionality $\mathcal{F}$, capturing all the properties required for the protocol and all the means of the adversary. In order to prove that a protocol Π emulates $\mathcal{F}$, one has to construct, for any polynomial adversary $\mathcal{A}$ (which controls the communication between the players), a simulator $\mathcal{S}$ such that no polynomial environment $\mathcal{Z}$ can distinguish between the real world (with the real players interacting with themselves and $\mathcal{A}$ and executing the protocol π) and the ideal world (with dummy players interacting with $\mathcal{S}$ and $\mathcal{F}$) with a significant advantage. The adversary can be either *adaptive*, i.e. allowed to corrupt users whenever it likes to, or *static*, i.e. required to choose which users to corrupt prior to the execution of the session sid of the protocol. After corrupting a player, $\mathcal{A}$ has complete access to the internal state and private values of the player, takes its entire control, and plays on its behalf. As this last possibility offers the widest set of challenges, we are going to only consider adaptive corruptions, while relying on reliable erasures so to wipe correctly the memory.

2.1.1 Simple UC Framework.

Canetti, Cohen and Lindell formalized a simpler variant in [CCL15], that we use here. This simplifies the description of the functionalities for the following reasons (in a nutshell): All channels are automatically assumed to be authenticated (as if we were working in the $\mathcal{F}_{\text{AUTH}}$-hybrid model); There is no need for *public delayed outputs* (waiting for the adversary before delivering a message to a party), neither for an explicit description of the corruptions. We refer the interested reader to [CCL15] for details.

2.2 Standard Cryptographic Primitives

2.2.1 Encryption

An encryption scheme $\mathcal{E}$ is described through four algorithms ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}$):

- $\mathsf{Setup}(1^{\mathfrak{K}})$, where $\mathfrak{K}$ is the security parameter, generates the global parameters param of the scheme;
- $\mathsf{KeyGen}(\mathsf{param})$ outputs a pair of keys, a (public) encryption key pk and a (private) decryption key dk;
- $\mathsf{Encrypt}(\mathsf{pk}, M; \rho)$ outputs a ciphertext $\mathbf{c} = \mathcal{C}(M)$, on the message M, under the encryption key pk, with the randomness ρ;
- $\mathsf{Decrypt}(\mathsf{dk}, \mathbf{c})$ outputs the plaintext M, encrypted in the ciphertext $\mathbf{c}$ or $\perp$.

Such encryption scheme is required to have the following security properties:

- *Correctness*: For every pair of keys $(\mathsf{ek}, \mathsf{dk})$ generated by KeyGen, every messages M, and every randomness ρ, we should have $\mathsf{Decrypt}(\mathsf{dk}, \mathsf{Encrypt}(\mathsf{ek}, M; \rho)) = M$.
- *Indistinguishability under Chosen Plaintext Attack [GM84]*: This notion ($\mathsf{IND}-\mathsf{CPA}$), formalized by the adjacent game, states that an adversary shouldn't be able to efficiently guess which message has been encrypted even if he chooses the two original plaintexts.
  The advantages are:

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E},\mathcal{A}}(\mathfrak{K}) = |\Pr[\mathsf{Exp}^{\mathsf{ind}-1}_{\mathcal{E},\mathcal{A}}(\mathfrak{K}) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ind}-0}_{\mathcal{E},\mathcal{A}}(\mathfrak{K}) = 1]|$$

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E}}(\mathfrak{K}, t) = \max_{\mathcal{A} \le t} \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E},\mathcal{A}}(\mathfrak{K}).$$

$\mathsf{Exp}^{\mathsf{ind}-b}_{\mathcal{E},\mathcal{A}}(\mathfrak{K})$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$
2. $(\mathsf{pk}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(M_0, M_1) \leftarrow \mathcal{A}(\texttt{FIND} : \mathsf{pk})$
4. $c^* \leftarrow \mathsf{Encrypt}(\mathsf{ek}, M_b)$
5. $b' \leftarrow \mathcal{A}(\texttt{GUESS} : c^*)$
6. $\texttt{RETURN}\ b'$

One might want to increase the requirements on the security of an encryption, in this case the $\mathsf{IND}-\mathsf{CPA}$ notion can be strengthened into Indistinguishability under Adaptive Chosen Ciphertext Attack $\mathsf{IND}-\mathsf{CCA2}$ (The non-adaptive notion was introduced in [NY90], while the adaptive one was introduced a year later in [RS92]):

- $\mathsf{IND}-\mathsf{CCA2}$: This notion states that an adversary should not be able to efficiently guess which message has been encrypted even if he chooses the two original plaintexts, and can ask several decryption of ciphertexts as long as they are not the challenge one.

$\mathsf{Exp}^{\mathsf{ind}-\mathsf{cca}-b}_{\mathcal{E},\mathcal{A}}(\mathfrak{K})$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$
2. $(\mathsf{pk}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(M_0, M_1) \leftarrow \mathcal{A}(\texttt{FIND} : \mathsf{pk}, \mathsf{ODecrypt}(\cdot))$
4. $c^* \leftarrow \mathsf{Encrypt}(\mathsf{ek}, M_b)$
5. $b' \leftarrow \mathcal{A}(\texttt{GUESS} : c^*, \mathsf{ODecrypt}(\cdot))$
6. $\texttt{IF } (c^*) \in \mathcal{CT} \texttt{ RETURN } 0$
7. $\texttt{ELSE RETURN } b'$

  - Where the $\mathsf{ODecrypt}$ oracle outputs the decryption of c under the challenge decryption key dk. The input queries (c) are added to the list $\mathcal{CT}$ of decrypted ciphertexts.

One may want to extend the notion of encryption to a labelled encryption, where the message M is encrypted but with some extra public information ℓ. This label can be useful to include session information for example.

2.2.2 Digital Signature

A digital signature scheme $\mathcal{S}$ [DH76, GMR88] allows a signer to produce a verifiable proof that he indeed produced a message. It is described through four algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verif})$:

Digital Signature Scheme
$\sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verif})$:
- $\mathsf{Setup}(1^{\mathfrak{K}})$ where $\mathfrak{K}$ is the security parameter, generates the global parameters param of the scheme, for example the message space;
- $\mathsf{KeyGen}(\mathsf{param})$, outputs a pair of $(\mathsf{sk}, \mathsf{vk})$, where sk is the (secret) signing key, and vk is the (public) verification key;
- $\mathsf{Sign}(\mathsf{sk}, M; \mu)$, outputs a signature $\sigma(M)$, on a message M, under the signing key sk, and some randomness μ;
- $\mathsf{Verif}(\mathsf{vk}, M, \sigma)$ checks the validity of the signature σ with respect to the message M and the verification key vk, and so outputs a bit.

In the following we will expect at least two properties for signatures:
- *Correctness*: For every pair $(\mathsf{vk}, \mathsf{sk})$ generated by KeyGen, for every message M, and for all randomness μ, we have $\mathsf{Verif}(\mathsf{vk}, M, \mathsf{Sign}(\mathsf{sk}, M; \mu)) = 1$.
- *Strong Existential Unforgeability under Chosen Message Attacks [SPMLS02].* Even after querying n valid signatures σ_i on chosen messages M_i, an adversary should not be able to output a fresh valid signature. To formalize this notion, we define a signing oracle $\texttt{Sign}$:
  - $\texttt{Sign}(\mathsf{vk}, m)$: This oracle outputs a signature on m valid under the verification key vk. The resulting pair (m, σ) is added to the signed pair set $\mathcal{S}'$.

$\mathsf{Exp}^{\mathsf{st}-\mathsf{uf}}_{\mathcal{S},\mathcal{A}}(\mathfrak{K})$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$
2. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(m^*, \sigma^*) \leftarrow \mathcal{A}(\mathsf{vk}, \texttt{Sign}(\mathsf{vk}, \cdot))$
4. $b \leftarrow \mathsf{Verif}(\mathsf{vk}, m^*, \sigma^*)$
5. $\texttt{IF } (m^*, \sigma^*) \in \mathcal{S}' \texttt{ RETURN } 0$
6. $\texttt{ELSE RETURN } b$

The probability of success against this game is denoted by

$$\mathsf{Succ}^{\mathsf{st}-\mathsf{uf}}_{\mathcal{S},\mathcal{A}}(\mathfrak{K}) = \Pr[\mathsf{Exp}^{\mathsf{st}-\mathsf{uf}}_{\mathcal{S},\mathcal{A}}(\mathfrak{K}) = 1], \quad \mathsf{Succ}^{\mathsf{st}-\mathsf{uf}}_{\mathcal{S}}(\mathfrak{K}, t) = \max_{\mathcal{A} \leq t} \mathsf{Succ}^{\mathsf{st}-\mathsf{uf}}_{\mathcal{S},\mathcal{A}}(\mathfrak{K}).$$

Or *Existential Unforgeability under Chosen Message Attacks [GMR88]* $(\mathsf{EUF}-\mathsf{CMA})$. Even after querying n valid signatures on chosen messages (M_i), an adversary should not be able to output a valid signature on a fresh message M. To formalize this notion, we define a signing oracle $\texttt{Sign}$:
  - $\texttt{Sign}(\mathsf{vk}, m)$: This oracle outputs a signature on m valid under the verification key vk. The requested message is added to the signed messages set $\mathcal{SM}$.

$\mathsf{Exp}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(\mathfrak{K})$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$
2. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(m^*, \sigma^*) \leftarrow \mathcal{A}(\mathsf{vk}, \texttt{Sign}(\mathsf{vk}, \cdot))$
4. $b \leftarrow \mathsf{Verif}(\mathsf{vk}, m^*, \sigma^*)$
5. $\texttt{IF } m^* \in \mathcal{SM} \texttt{ RETURN } 0$
6. $\texttt{ELSE RETURN } b$

The probability of success against this game is denoted by

$$\mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(\mathfrak{K}) = \Pr[\mathsf{Exp}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(\mathfrak{K}) = 1], \quad \mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S}}(\mathfrak{K}, t) = \max_{\mathcal{A} \leq t} \mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(\mathfrak{K}).$$

$\star$

Part I

Building Hash proof System

Chapter 3

Hash Proof Definition

Contents

3.1 Definition

Smooth Projective Hash Functions were introduced by Cramer and Shoup [CS02]. They allow to evaluate a function of an input using two different ways, either using a trapdoor corresponding to the public key of the function, or using a witness showing that the input fulfills some properties. Of course, the evaluation should match if and only if the trapdoor is the valid one, and the input does indeed fulfills the property for the said witness.

More formally, a projective hashing family is a family of hash functions that can be evaluated in two ways: using the (secret) hashing key, one can compute the function on every point in its domain, whereas using the (public) *projected* key one can only compute the function on a special subset of its domain. Such a family is deemed *smooth* if the value of the hash function on any point outside the special subset is independent of the projected key. The notion of SPHF has found applications in various contexts in cryptography (*e.g.* [GL03, Kal05, ACP09]):

Smooth Projective Hashing System
⌜ A Smooth Projective Hash Function over a language $\mathscr{L} \subset X$, onto a set $\mathbb{H}$, is defined by five algorithms $(\mathsf{Setup}, \mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$:

- $\mathsf{Setup}(1^{\mathfrak{K}})$ where $\mathfrak{K}$ is the security parameter, generates the global parameters param of the scheme, and the description of an $\mathcal{NP}$ language $\mathscr{L}$;
- $\mathsf{HashKG}(\mathscr{L}, \mathsf{param})$, outputs a hashing key hk for the language $\mathscr{L}$;
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathscr{L}, \mathsf{param}), W)$, derives the projection key hp, possibly depending on the word W [GL03, ACP09] thanks to the hashing key hk.
- $\mathsf{Hash}(\mathsf{hk}, (\mathscr{L}, \mathsf{param}), W)$, outputs a hash value $v \in \mathbb{H}$, thanks to the hashing key hk, and W
- $\mathsf{ProjHash}(\mathsf{hp}, (\mathscr{L}, \mathsf{param}), W, w)$, outputs the hash value $v' \in \mathbb{H}$, thanks to the projection key hp and the witness w that $W \in \mathscr{L}$.

⌟

In the following, we consider $\mathscr{L}$ as a hard-partitioned subset of X, i.e. it is computationally hard to distinguish a random element in $\mathscr{L}$ from a random element in $X \setminus \mathscr{L}$.

A Smooth Projective Hash Function SPHF should satisfy the following properties:

- *Correctness*: Let $W \in \mathscr{L}$ and w a witness of this membership. Then, for all hashing keys hk and associated projection keys hp we have $\mathsf{Hash}(\mathsf{hk}, (\mathscr{L}, \mathsf{param}), W) = \mathsf{ProjHash}(\mathsf{hp}, (\mathscr{L}, \mathsf{param}), W, w)$.

- *Smoothness*: For all $W \in X \setminus \mathscr{L}$ the following distributions are statistically indistinguishable:

$$\Delta_0 = \left\{ (\mathscr{L}, \mathsf{param}, W, \mathsf{hp}, v) \;\middle|\; \begin{array}{l} \mathsf{param} = \mathsf{Setup}(1^{\mathfrak{K}}), \mathsf{hk} = \mathsf{HashKG}(\mathscr{L}, \mathsf{param}), \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathscr{L}, \mathsf{param}), W), v = \mathsf{Hash}(\mathsf{hk}, (\mathscr{L}, \mathsf{param}), W) \end{array} \right\}$$

$$\Delta_1 = \left\{ (\mathscr{L}, \mathsf{param}, W, \mathsf{hp}, v) \;\middle|\; \begin{array}{l} \mathsf{param} = \mathsf{Setup}(1^{\mathfrak{K}}), \mathsf{hk} = \mathsf{HashKG}(\mathscr{L}, \mathsf{param}), \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathscr{L}, \mathsf{param}), W), v \xleftarrow{\$} \mathbb{H} \end{array} \right\}.$$

  This is formalized by

$$\mathsf{Adv}^{\mathsf{smooth}}_{\mathrm{SPHF}}(\mathfrak{K}) = \sum_{V \in \mathbb{H}} \left| \Pr_{\Delta_1}[v = V] - \Pr_{\Delta_0}[v = V] \right| \text{ is negligible.}$$

- *Pseudo-Randomness*: If $W \in \mathscr{L}$, then without a witness of membership the two previous distributions should remain computationally indistinguishable: for any adversary $\mathcal{A}$ within reasonable time

$$\mathsf{Adv}^{\mathsf{pr}}_{\mathrm{SPHF},\mathcal{A}}(\mathfrak{K}) = |\Pr_{\Delta_1}[\mathcal{A}(\mathscr{L}, \mathsf{param}, W, \mathsf{hp}, v) = 1] - \Pr_{\Delta_0}[\mathcal{A}(\mathscr{L}, \mathsf{param}, W, \mathsf{hp}, v) = 1]| \text{ is negligible.}$$

Pseudo-Randomness comes directly from the Smoothness and the Hard-Partitioned Subset.

Abdalla *et al.* [ACP09] explained how to combine SPHF to deal with conjunctions and disjunctions of the languages. In the following we simply recall those results:

Let us assume we have two Smooth Projective Hash Functions, defined by SPHF_1 and SPHF_2, on two languages, $\mathscr{L}_1$ and $\mathscr{L}_2$ respectively, both subsets of X, with hash values in the same group $(\mathbb{H}, \oplus)$. We note W an element of X, w_i a witness that $W \in \mathscr{L}_i$, $\mathsf{hk}_i = \mathsf{HashKG}_i(\mathscr{L}_i, \mathsf{param})$ and $\mathsf{hp}_i = \mathsf{ProjKG}_i(\mathsf{hk}_i, (\mathscr{L}_i, \mathsf{param}_i), W)$.

We can then define the SPHF on $\mathscr{L} = \mathscr{L}_1 \cap \mathscr{L}_2$, where $w = (w_1, w_2)$ as:

- $\mathsf{Setup}(1^{\mathfrak{K}})$, $\mathsf{param} = (\mathsf{param}_1, \mathsf{param}_2)$, and $\mathscr{L} = \mathscr{L}_1 \cap \mathscr{L}_2$;
- $\mathsf{HashKG}(\mathscr{L}, \mathsf{param})$: $\mathsf{hk} = (\mathsf{hk}_1, \mathsf{hk}_2)$
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathscr{L}, \mathsf{param}), W)$: $\mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2)$
- $\mathsf{Hash}(\mathsf{hk}, (\mathscr{L}, \mathsf{param}), W)$: $\mathsf{Hash}_1(\mathsf{hk}_1, (\mathscr{L}_1, \mathsf{param}_1), W) \oplus \mathsf{Hash}_2(\mathsf{hk}_2, (\mathscr{L}_2, \mathsf{param}_2), W)$
- $\mathsf{ProjHash}(\mathsf{hp}, (\mathscr{L}, \mathsf{param}), W, w = (w_1, w_2))$:

$$\mathsf{ProjHash}_1(\mathsf{hp}_1, (\mathscr{L}_1, \mathsf{param}_1), W, w_1) \oplus \mathsf{ProjHash}_2(\mathsf{hp}_2, (\mathscr{L}_2, \mathsf{param}_2), W, w_2)$$

We can also define the SPHF on $\mathscr{L} = \mathscr{L}_1 \cup \mathscr{L}_2$, where $w = w_1$ or $w = w_2$ as:

- $\mathsf{Setup}(1^{\mathfrak{K}})$, $\mathsf{param} = (\mathsf{param}_1, \mathsf{param}_2)$, and $\mathscr{L} = \mathscr{L}_1 \cup \mathscr{L}_2$;
- $\mathsf{HashKG}(\mathscr{L}, \mathsf{param})$: $\mathsf{hk} = (\mathsf{hk}_1, \mathsf{hk}_2)$
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathscr{L}, \mathsf{param}), W)$: $\mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2, \mathsf{hp}_\Delta)$ where

$$\mathsf{hp}_\Delta = \mathsf{Hash}_1(\mathsf{hk}_1, (\mathscr{L}_1, \mathsf{param}_1), W) \oplus \mathsf{Hash}_2(\mathsf{hk}_2, (\mathscr{L}_2, \mathsf{param}_2), W)$$

- $\mathsf{Hash}(\mathsf{hk}, (\mathscr{L}, \mathsf{param}), W)$: $\mathsf{Hash}_1(\mathsf{hk}_1, (\mathscr{L}_1, \mathsf{param}_1), W)$
- $\mathsf{ProjHash}(\mathsf{hp}, (\mathscr{L}, \mathsf{param}), W, w)$: If $W \in \mathscr{L}_1$, $\mathsf{ProjHash}_1(\mathsf{hp}_1, (\mathscr{L}_1, \mathsf{param}_1), W, w_1)$, else (if $W \in \mathscr{L}_2$), $\mathsf{hp}_\Delta \ominus \mathsf{ProjHash}_2(\mathsf{hp}_2, (\mathscr{L}_2, \mathsf{param}_2), W, w_2)$

Remark It should be noted, that while this construction of an SPHF for $\mathscr{L}_1 \cup \mathscr{L}_2$ is correct, it can not always be used in a black-box way for some applications, because a dishonest verifier could provide an invalid hp_Δ without being detected and be able to secretly test if a word belongs to $\mathscr{L}_1$.

3.2 Various Subtypes

Observing the various types of SPHF existing in the literature, we provided a classification of Smooth Projective Hash Function in [BBC+13b] named after the authors who used them in the given way for the first time.

3.2.1 GL-SPHF

Those SPHF are named after [GL03], they correspond to the basic definition of an SPHF. The word W is used as an input of the ProjKG function, and they remain smooth for any word outside the language.

The smoothness requires the following distributions to be close.

$$\Delta_0 = \left\{ (\mathsf{hp}, v) \;\middle|\; \mathsf{hk} = \mathsf{HashKG}(\mathscr{L}), \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, \mathscr{L}, W), v = \mathsf{Hash}(\mathsf{hk}, \mathscr{L}, W) \right\}$$
$$\Delta_1 = \left\{ (\mathsf{hp}, v) \;\middle|\; \mathsf{hk} = \mathsf{HashKG}(\mathscr{L}), \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, \mathscr{L}, W), v \xleftarrow{\$} \mathbb{H} \right\}.$$

Such SPHFs seem to be the baseline. If a language is manageable with an SPHF, then there exists a corresponding GL-SPHF. This ease of construction has however a drawback, as the projection key is dependent on the word, this often induces an extra flow in the protocols.

	ElGamal	Cramer Shoup
W	$h^r M, g^r$	$h^r M, g_1^r, g_2^r, (cd^\chi)^r$
hk	$\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p$	$\alpha, \beta, \gamma, \delta \xleftarrow{\$} \mathbb{Z}_p$
hp	$h^\alpha g^\beta$	$h^\alpha g_1^\beta g_2^\gamma (cd^\chi)^\delta$
v	$(W_1/M)^\alpha W_2^\beta$	$(W_1/M)^\alpha W_2^\beta W_3^\gamma W_4^\delta$

Figure 3.1: GL-SPHF over the language of valid encryption of M

One should pay attention to the fact that $\chi = \mathcal{H}(W_1, W_2, W_3, \ldots)$ so hp is dependent on W.

3.2.2 CS-SPHF

Those SPHF are named after [CS02] as the original definition is close to the one we use to define this category [1]. On one hand, the projection key is constructed before seeing the word W, on the other the word W has to be independent from the projection key.

This is described by the following distributions:

$$\Delta_0 = \left\{ (\mathsf{hp}, v) \;\middle|\; \mathsf{hk} = \mathsf{HashKG}(\mathscr{L}), \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, \mathscr{L}, \perp), v = \mathsf{Hash}(\mathsf{hk}, \mathscr{L}, W) \right\}$$
$$\Delta_1 = \left\{ (\mathsf{hp}, v) \;\middle|\; \mathsf{hk} = \mathsf{HashKG}(\mathscr{L}), \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, \mathscr{L}, \perp), v \xleftarrow{\$} \mathbb{H} \right\}.$$

This category is here for "historical" reasons. There is no apparent benefit from constructing such SPHF, because on one hand, the projection key has to be constructed without knowing the word which makes it hard to do, but the security only considers words chosen non-adaptively by the adversary which is underwhelming.

3.2.3 KV-SPHF

This last category is named after [KV11]. This considers SPHF where the projection key is independent from the word, and where the smoothness need to hold even for an adaptively chosen word $W = f(\mathsf{hp})$.

$$\Delta_0 = \left\{ (\mathsf{hp}, v) \;\middle|\; \mathsf{hk} = \mathsf{HashKG}(\mathscr{L}), \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, \mathscr{L}, \perp), v = \mathsf{Hash}(\mathsf{hk}, \mathscr{L}, f(\mathsf{hp})) \right\}$$
$$\Delta_1 = \left\{ (\mathsf{hp}, v) \;\middle|\; \mathsf{hk} = \mathsf{HashKG}(\mathscr{L}), \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, \mathscr{L}, \perp), v \xleftarrow{\$} \mathbb{H} \right\}.$$

Such SPHF are harder to build than GL-SPHF, however due to the independence of the projection key and the security holds even in the presence of adaptively chosen words, this helps to reduce the number of round involved in a protocol of allow synchronous flows.

While [KV11] builds such SPHF over a CPA language (enhanced to CCA with Naor Yung [NY90] transform), it has been an open problem to build one over a CCA-2 scheme *à la* Cramer Shoup until our result from [BBC+13b]. It is not clear whether all languages manageable via a GL-SPHF can also

	ElGamal	Cramer Shoup
W	$h^r M, g^r$	$h^r M, g_1^r, g_2^r, (cd^\chi)^r$
hk	$\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p$	$\alpha_1, \alpha_2, \beta, \gamma, \delta \xleftarrow{\$} \mathbb{Z}_p$
hp	$h^\alpha g^\beta$	$h^{\alpha_1} g_1^\beta g_2^\gamma c^\delta, h^{\alpha_2} d^\delta$
v	$(W_1/M)^\alpha W_2^\beta$	$(W_1/M)^{\alpha_1+\chi\alpha_2} W_2^\beta W_3^\gamma W_4^\delta$

Figure 3.2: KV-SPHF over the language of valid encryption of M

be handled through a KV-SPHF. As a rule of thumb, as long as a language is not quadratic (or worse), then there is a way to build a KV-SPHF.

One can see that the SPHF for Cramer Shoup is a little more involved in this case, the projection key requires 2 elements instead of 1, this split was the key in producing a projection key hp not requiring the knowledge of W (as χ is only used in the final hash computation).

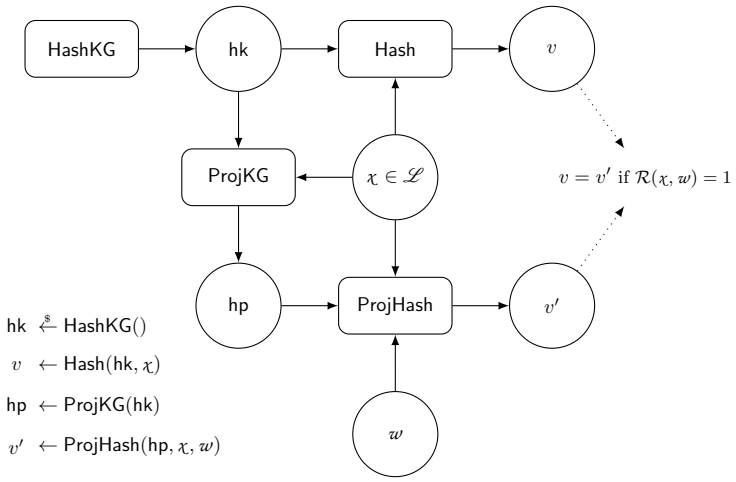


Figure 3.3: Synthetic view of a (GL) SPHF evaluation

⋆

[1] To nitpick, the original definition considers the smoothness for a random word outside the language, while ours holds for any word outside the language

Chapter 4

Languages and Underlying Hypotheses

Contents

SPHF are built for a given language. A common result with SPHF is that contrarily to Zero-Knowledge, there exist languages in NP that cannot be handled by an SPHF [GGSW13][1], in other words, there exists languages where there is no secure SPHF whose sole witness are the solution to a given problem.

4.1 Side Result: SPHF for an encryption of a solution of an NP problem

This is not a huge constraint however, because one can easily show that assuming the existence of an encryption scheme E manageable with an SPHF so there exists a SPHF $\mathcal{S}_E$ for the language $\mathscr{L}_E$, then one can build an SPHF for the language of encryption of a solution of the problem.

This has never been explicitly detailed, hence we are going to sketch the idea of this technique, which is similar to the original proof to show there exist a Zero-Knowledge Proof for all NP. For this we consider a 3-coloring instance. 3-coloring being NP-Complete this would lead to the conclusion. For every edge in the graph, we are going to build a SPHF proving the linked nodes are different. This can be done in two ways:

- Either one can check the validity of the computation of the Projective hash computed for $\mathcal{S}_E$ over $\mathscr{L}_E$. In this case, one can use the methodology described in [BCV15], recalled 4.5, page 25, to build a SPHF that *fails* to prove the the join node are identical.
- Or, one has to take the long way round, and build an SPHF for the disjunction of the various cases ((Color 1 AND Color 2) OR (Color 1 and Color 3) OR ... OR (Color 3 and Color 2))

One can then build an SPHF for the conjunction of those inner SPHF (one for each edge) together with a conjunction SPHF for the language "This is an encryption of a valid color" (again this can be done

[1] SPHF leading to trivial Witness Encryption, this would mean Witness Encryption for all NP which requires a collapse of the polynomial hierarchy

with a disjunction of cases (Color 1 OR Color 2 OR Color 3)) which leads to the conclusion. Because, under the smoothness of the various inner SPHF, the final one is also smooth.

This emphasizes the fact, that, in addition to being natural, building an SPHF compatible with a commitment seems to be the best way to design hash proof systems instead of tackling languages directly.

4.2 Elliptic Curves

In [EHK+13], the authors introduced the matrix notation to generalize constructions over elliptic curves. This fits really well with most (if not all) known constructions of SPHF, and allowed us to introduce **Structure Preserving Smooth Projective Hash Functions** [BC16].

4.2.1 Reminder on Matrix Notation

Let GGen be a probabilistic polynomial time (PPT) algorithm that on input $1^{\mathfrak{K}}$ returns a description $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ of asymmetric pairing groups where $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ are cyclic groups of order p for a $\mathfrak{K}$-bit prime p, g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and $e : \mathbb{G}_1 \times \mathbb{G}_2$ is an efficiently computable (non-degenerate) bilinear map. Define $g_T := e(g_1, g_2)$, which is a generator in $\mathbb{G}_T$.

We use implicit representation of group elements as introduced in [EHK+13]. For $s \in \{1, 2, T\}$ and $a \in \mathbb{Z}_p$ define $[a]_s = g_s^a \in \mathbb{G}_s$ as the *implicit representation* of a in $\mathbb{G}_s$. More generally, for a matrix $\mathbf{A} = (a_{ij}) \in \mathbb{Z}_p^{n\times m}$ we define $[\mathbf{A}]_s$ as the implicit representation of $\mathbf{A}$ in $\mathbb{G}_s$:

$$[\mathbf{A}]_s := \begin{pmatrix} g_s^{a_{11}} & \dots & g_s^{a_{1m}} \\ & & \\ g_s^{a_{n1}} & \dots & g_s^{a_{nm}} \end{pmatrix} \in \mathbb{G}_s^{n\times m}$$

We will always use this implicit notation of elements in $\mathbb{G}_s$, i.e., we let $[a]_s \in \mathbb{G}_s$ be an element in $\mathbb{G}_s$. Note that from $[a]_s \in \mathbb{G}_s$ it is generally hard to compute the value a (discrete logarithm problem in $\mathbb{G}_s$). Further, from $[b]_T \in \mathbb{G}_T$ it is hard to compute the value $[b]_1 \in \mathbb{G}_1$ and $[b]_2 \in \mathbb{G}_2$ (pairing inversion problem). Obviously, given $[a]_s \in \mathbb{G}_s$ and a scalar $x \in \mathbb{Z}_p$, one can efficiently compute $[ax]_s \in \mathbb{G}_s$. Further, given $[a]_1, [b]_2$ one can efficiently compute $[ab]_T$ using the pairing e. For $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^k$ define $e([\mathbf{a}]_1, [\mathbf{b}]_2) := [\mathbf{a}^\top \mathbf{b}]_T \in \mathbb{G}_T$. We recall the definition of the matrix Diffie-Hellman (MDDH) assumption [EHK+13].

Matrix Distribution

⌜ Let $k \in \mathbb{N}$. We call $\mathcal{D}_k$ a matrix distribution if it outputs matrices in $\mathbb{Z}_p^{(k+1)\times k}$ of full rank k in polynomial time. ⌟

Without loss of generality, we assume the first k rows of $\mathbf{A} \xleftarrow{\$} \mathcal{D}_k$ form an invertible matrix, we denote this matrix $\overline{\mathbf{A}}$, while the last line is denoted $\underline{A}$. The $\mathcal{D}_k$-Matrix Diffie-Hellman problem is to distinguish the two distributions $([\mathbf{A}], [\mathbf{Aw}])$ and $([\mathbf{A}], [\mathbf{u}])$ where $\mathbf{A} \xleftarrow{\$} \mathcal{D}_k$, $\mathbf{w} \xleftarrow{\$} \mathbb{Z}_p^k$ and $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$.

$\mathcal{D}_k$-Matrix Diffie-Hellman Assumption $\mathcal{D}_k$-MDDH

⌜ Let $\mathcal{D}_k$ be a matrix distribution and $s \in \{1, 2, T\}$. We say that the $\mathcal{D}_k$-Matrix Diffie-Hellman ($\mathcal{D}_k$-MDDH) Assumption holds relative to GGen in group $\mathbb{G}_s$ if for all PPT adversaries $\mathcal{D}$,

$$\mathbf{Adv}_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{D}) := |\Pr[\mathcal{D}(\mathcal{G}, [\mathbf{A}]_s, [\mathbf{Aw}]_s) = 1] - \Pr[\mathcal{D}(\mathcal{G}, [\mathbf{A}]_s, [\mathbf{u}]_s) = 1]| = \mathsf{negl}(\lambda),$$

where the probability is taken over $\mathcal{G} \xleftarrow{\$} \mathsf{GGen}(1^\lambda)$, $\mathbf{A} \xleftarrow{\$} \mathcal{D}_k, \mathbf{w} \xleftarrow{\$} \mathbb{Z}_p^k, \mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$. ⌟

For each $k \geq 1$, [EHK+13] specifies distributions $\mathcal{L}_k, \mathcal{U}_k, \dots$ such that the corresponding $\mathcal{D}_k$-MDDH assumption is the k-Linear assumption, the k-uniform and others. All assumptions are generically secure in bilinear groups and form a hierarchy of increasingly weaker assumptions. For $k = 2$, where $a_1, \dots, a_6 \xleftarrow{\$} \mathbb{Z}_p$, we have:

$$\mathcal{L}_2 : \mathbf{A} = \begin{pmatrix} a_1 & 0 \\ 0 & a_2 \\ 1 & 1 \end{pmatrix} \quad \mathcal{U}_2 : \mathbf{A} = \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \\ a_5 & a_6 \end{pmatrix}.$$

HPS and beyond

Building a hash proof system for Matrix Diffie Hellman languages is natural. The languages often consider whether a word W lives in an affine space of the form $[\mathbf{Ar} + \mathbf{b}]_*$ for given $[\mathbf{A}, \mathbf{b}]_*$.

In these cases, building the corresponding hash proof system boils down to sampling a appropriately sized scalar vector hk, publishing $\mathsf{hp} = [\mathsf{hk} \cdot \mathbf{A}]_*$, on one hand the hashing algorithm computes $v = [\mathsf{hk} \cdot (W - \mathbf{b})]_*$, while the projected hashing algorithm simply computes $v' = [\mathsf{hp} \cdot \mathbf{r}]_*$.

	ElGamal	Cramer Shoup
W	$[hr, r]$	$[hr, r, g_2 r, (c + \chi d)r]$
hk	$\boldsymbol{\alpha} \xleftarrow{\$} \mathbb{Z}_p^{1\times 2}$	$\boldsymbol{\alpha} \xleftarrow{\$} \mathbb{Z}_p^{1\times 4}$
hp	$[\boldsymbol{\alpha}(h, 1)^\top]$	$[\boldsymbol{\alpha}(h, 1, g_2, c + \chi d)^\top]$
v	$[\boldsymbol{\alpha} W]$	$[\boldsymbol{\alpha} W]$

Figure 4.1: GL-SPHF over the language of valid encryption of $1_{\mathbb{G}}$ in matrix notation

4.2.2 Languages *à la* Groth Sahai

In [GS08], Groth and Sahai proposed a methodology to build efficient non-interactive zero-knowledge proofs of knowledge in the standard model for pairing product equations.

In their definition a pairing product equation is an equation of the form:

$$[\sum_i \mathcal{X}_i \mathcal{B}_i + \sum_j \mathcal{A}_j \mathcal{Y}_j + \sum_i \sum_j \mathcal{X}_i \mathcal{Y}_j \gamma_{ij}]_T = \sum_k [\mathcal{A}_k \mathcal{B}_k]_T$$

Where, $[\mathcal{A}_*]_1$ are public elements in $\mathbb{G}_1$, $[\mathcal{B}_*]_2$ in $\mathbb{G}_2$, γ_* are public scalars while $[\mathcal{X}_*]_1$, $[\mathcal{Y}_*]_2$ are unknown in $\mathbb{G}_1$ (resp $\mathbb{G}_2$).

The Groth Sahai methodology can also work for scalar equation or multi exponent equation (ie where one side of the unknowns are in a group, and the other are scalars).

In [BBC+13a], we have shown that Smooth Projective Hash Functions can be used to handle the same kind of languages, and in fact a little bit more:

$$[\sum_i \mathcal{X}_i \mathcal{B}_i + \sum_j \mathcal{A}_j \mathcal{Y}_j + \sum_i \sum_j \mathcal{X}_i \mathcal{Y}_j \gamma_{ij}]_T = \sum_k [C_k z_k]_T + \sum_i \sum_j \sum_\ell [\mathcal{X}_i \mathcal{B}_j z_{ij\ell} + \mathcal{A}_i \mathcal{Y}_j z'_{ij\ell}]_T + \sum_\ell [Z_\ell]_T$$

Where, $[\mathcal{A}_*]_1$ are public elements in $\mathbb{G}_1$, $[\mathcal{B}_*]_2$ in $\mathbb{G}_2$, γ_* are public scalars while $[\mathcal{X}_*]_1$, $[\mathcal{Y}_*]_2$, $[\mathcal{Z}_*]_T$ are unknowns in $\mathbb{G}_1$ (resp $\mathbb{G}_2$, $\mathbb{G}_T$), and z_* unknown scalars in $\mathbb{Z}_p$.

The construction rely on the fact that a set of $\mathcal{X}, \mathcal{Y}, \mathcal{Z}, z$ is a solution if and only if every terms cancels out nicely so we are going to commit to every single variable with a linearly homomorphic commitment, build SPHF for the language of "Valid encryption of a variable" while forcing that the part of the hash key working directly on the encryption part carrying the variable ($h^r M$ in ElGamal for instance) is the same for every SPHF. This way if everything is indeed a valid encryption, one would obtain the left side of the equation raised to this value, and so we simply need to check whether this is equal to the right side raised to same one.

For example, let us consider the equation $e(f, \mathcal{Y}_1) \cdot e(g, \mathcal{Y}_2) = 1_T$, that checks whether someone knows two elements in $\mathbb{G}_2$ with the same discrete logarithms that two in $\mathbb{G}_1$. [2]

Our methodology would proceed as follows:

- First commit to both variables: $W_1 = [hr + \mathcal{Y}_1, r]_2, W_2 = [hs + \mathcal{Y}_2, s]_2$
- Then the verifier runs $\mathsf{HashKG}, \mathsf{ProjKG}$: $\lambda, \gamma_1, \gamma_2 \xleftarrow{\$} \mathbb{Z}_p^*$, set $\mathsf{hk}_1 = (\lambda, \gamma_1), \mathsf{hk}_2 = (\lambda, \gamma_2)$, $\mathsf{hp}_1 = [h\lambda + \gamma_1]_2, \mathsf{hp}_2 = [h\lambda + \gamma_2]_2$
- The user runs $\mathsf{ProjHash}(\mathsf{hp}, (r, s))$ and obtains $v' = [f\mathsf{hp}_1 r + \mathsf{hp}_2 s]_T$
- While the verifier runs $\mathsf{Hash}(\mathsf{hk}, W) : v = [f\mathsf{hk}_1 W_1 + \mathsf{hk}_2 W_2]_T = v' + [f\lambda\mathcal{Y}_1 + \lambda\mathcal{Y}_2]_T$

Those two values are equals if and only if the equation is fulfilled. It should be noted, that as long as there is no quadratic term, the underlying SPHF is KV, allowing the verifier to compute (and publish) the projection key before the interaction.

[2] If one does not forbid the trivial case $\mathcal{Y}_1 = \mathcal{Y}_2 = 0$, this equation has no particular merit besides being a simple example.

	SPHF	SP-SPHF
MDDH	$[\mathbf{Br}]$	$[\mathbf{Br}]_1$
Witness w	r	$[r]_2$
hk	$\boldsymbol{\lambda} \xleftarrow{\$} \mathbb{Z}_p^{1\times k}$	$\boldsymbol{\lambda} \xleftarrow{\$} \mathbb{Z}_p^{1\times k}$
hp	$[\mathsf{hp} = \boldsymbol{\lambda}\cdot\mathbf{B}]$,	$[\mathsf{hp} = \boldsymbol{\lambda}\cdot\mathbf{B}]_1$
$\mathsf{Hash}(\mathsf{hk}, \mathbf{u})$	$[\boldsymbol{\lambda}\cdot\mathbf{W}]$	$[\boldsymbol{\lambda}\cdot\mathbf{W}]_T$
$\mathsf{ProjHash}(\mathsf{hp}, \mathbf{u}, w)$	$[\mathsf{hp}\cdot\mathbf{r}]$	$[\mathsf{hp}\cdot\mathbf{r}]_T$

Figure 4.2: Example of (naive) conversion of SPHF into SP-SPHF (matricial notations)

4.2.3 Structure Preserving Smooth Projective Hash Function

Smooth Projective Hash Functions are not Zero-Knowledge, however they end up being often more efficient. In [BC16], we aimed at obtaining the best of both worlds by allowing Hash Proof System to use Zero Knowledge Proofs as witnesses. In other world, contrary to what was currently done instead of using a scalar as a group element, one could now use directly group elements, like for example Groth Sahai proofs of knowledge

When using SPHF with commitments to achieve an implicit decommitment, the language is usually defined on group elements, with projection keys being group elements, and witnesses being scalars. While in several applications, this has already lead to efficient constructions, the fact that witnesses have to be scalars (and in the particular case of commitments, the randomness used to commit) leads to drastic restrictions when trying to build protocols secure against adaptive corruptions in the UC framework.

In the same spirit as Structure-Preserving Signature [AFG+10], we proposed the notion of Structure-Preserving Smooth Projective Hash Functions (SP-SPHF), where both words, witnesses and projection keys are group elements, and hash and projective hash computations are doable with pairings in the context of bilinear groups.

We show how to transform every previously known pairing-less construction of SPHF to fit this methodology, and then propose several applications in which storing a group element as a witness allows to avoid the drastic restrictions that arise when building protocols secure against adaptive corruptions in the UC framework with a scalar as witness. Asking the witness to be a group element enables us to gain more freedom in the simulation (the discrete logarithm of this element and / or real extraction from a commitment). For instance, the simulator can always commit honestly to a random message, since it only needs to modify its witness in the equivocation phase. Furthermore, it allows to avoid bit-per-bit construction. Such design carries similarity with the publicly verifiable MACs from [KPW15], where the pairing operation allows to relax the verification procedure.

Structure-Preserving Smooth Projective Hash Functions

A Structure-Preserving Smooth Projective Hash Function over a language $\mathscr{L} \subset X$ onto a set $\mathcal{H}$ is defined by 4 algorithms ($\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash}$):

- $\mathsf{HashKG}(\mathscr{L}, \mathsf{param})$, outputs a hashing key hk for the language $\mathscr{L}$;
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathscr{L}, \mathsf{param}), W)$, derives the projection key hp thanks to the hashing key hk.
- $\mathsf{Hash}(\mathsf{hk}, (\mathscr{L}, \mathsf{param}), W)$, outputs a hash value $H \in \mathcal{H}$, thanks to the hashing key hk, and W
- $\mathsf{ProjHash}(\mathsf{hp}, (\mathscr{L}, \mathsf{param}), W, w)$, outputs the value $H' \in \mathcal{H}$, thanks to hp and the witness w that $W \in \mathscr{L}$.

Remark We stress that, contrarily to classical SPHF, both hp, W and more importantly w are base group elements, and so live in the same space.

Of course, one can again derive the three previous subtypes KV, GL, CS depending on when the word W is sampled and used in the algorithm.

It should be noted that converting an SPHF to an SP-SPHF is easily done, and does not weaken the subgroup decision assumption (k-MDDH in the following) linked to the original language.

We give in Figure 4.2, page 20 an example of conversion from the SPHF over a valid MDDH to an SP-SPHF for such a tuple.

One can see two things. Besides adding a pairing the transformation does not weaken the underlying security. More importantly, the general structure is the same, hence as we have shown in [BC16], there is no drawbacks in using group elements as witnesses, and the smoothness is going to be proven in the same way.

4.3 Euclidean Lattices

Post Quantum Cryptography opens new play fields for Cryptographers to instantiate classical tools.

One of the most promising family of problems is around Learning with Error (LWE).

Until our result, there was only one construction of SPHF for a lattice-based encryption scheme in the standard model, given by Katz and Vaikuntanathan [KV09]. There was also a subsequent work by Zhang and Yu who propose an interesting new lattice-based SPHF in [ZY17]. But the language of the SPHF relies on simulation-sound non-interactive zero-knowledge proofs which we do not know how to construct solely under lattice-based assumptions without random oracle.

Unfortunately, the only standard-model lattice-based SPHF construction in [KV09] has a main drawback: the language of the SPHF is not simply defined as the set of valid standard LWE ciphertexts. Naturally, the set of valid ciphertexts of 0 should correspond to the set of ciphertexts close to the lattice defined by the public key. Instead, their language includes all the ciphertexts $\mathbf{c}$ such that at least one integer multiple is close to the public lattice. This makes the decryption procedure very costly (about q trapdoor inversions), and forbids the use of super-polynomial modulus q. This limitation is a serious obstacle to the construction of a stronger type of SPHF introduced in [KV11], namely *word-independent* SPHF for which the *projection key* (which can be seen as the public key of the SPHF) does not depend on the ciphertext $\mathbf{c}$ (a.k.a., word in the SPHF terminology).[3]

We proposed to tackle this issue in [BBDQ18]. Here is a short technical overview of our main contribution, namely the constructions of new lattice-based SPHFs.

We focus on the language of dual-Regev ciphertexts $\mathbf{c}$ of 0: $\mathbf{c} = \mathbf{A}\mathbf{s} + \mathbf{e} \in \mathbb{Z}_q^m$, where $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ is a public matrix, while $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mathbf{e} \in \mathbb{Z}_q^m$ correspond to the randomness of the ciphertext. The vector $\mathbf{e}$ is supposed to be small, i.e., $\mathbf{c}$ is close to the q-ary lattice Λ generated by $\mathbf{A}$.

Intuitively, an SPHF allows a prover knowing $\mathbf{s}$ and $\mathbf{e}$ to prove to a verifier that $\mathbf{c}$ is indeed a ciphertext of 0. The naive and natural construction works as follows (of course we only obtain a bit-SPHF, classical techniques can then be used to a full-fledged SPHF). The verifier generates a *small* random vector $\mathsf{hk} = \mathbf{h} \in \mathbb{Z}_q^m$ called a *hashing key*. It then "hashes" the ciphertext into a *hash value* $H = R(\langle \mathbf{h}, \mathbf{c} \rangle) \in \{0, 1\}$, where R is a *rounding function* from $\mathbb{Z}_q$ to $\{0, 1\}$ to be chosen later. The verifier also derives from $\mathsf{hk} = \mathbf{h}$, a *projection key* $\mathsf{hp} = \mathbf{p} = \mathbf{A}^t\mathbf{h} \in \mathbb{Z}_q^n$ that it sends to the prover. The prover can then compute the *projected hash value* $H' = R(\langle \mathbf{p}, \mathbf{s} \rangle)$ from the projection key $\mathbf{p}$ and the randomness of the ciphertext $\mathbf{s}$ and $\mathbf{e}$. It can send this projected hash value to the verifier which will accept the proof, if H' matches its hash value H.

We remark that if indeed $\mathbf{c} = \mathbf{A}\mathbf{s} + \mathbf{e}$ with $\mathbf{e}$ small enough (recall that $\mathbf{h}$ is small as well):

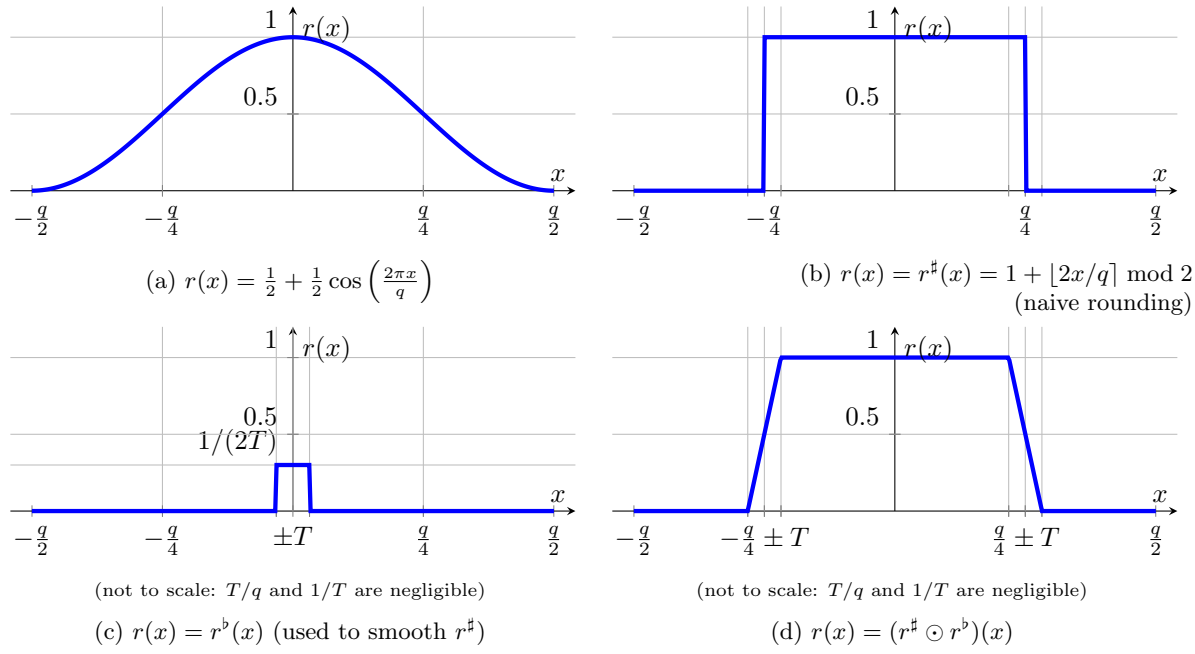
$$\langle \mathbf{h}, \mathbf{c} \rangle = \mathbf{h}^t\mathbf{A}\mathbf{s} + \mathbf{h}^t\mathbf{e} \approx \mathbf{h}^t\mathbf{A}\mathbf{s} = \langle \mathbf{p}, \mathbf{s} \rangle \ .$$

Hence, if R is carefully chosen, we can ensure that with high probability (e.g., at least $3/4$), $H = H'$, and the verifier will accept the prover's "proof." This property is called *approximate correctness*. An SPHF also needs to satisfy a security property to be useful, called *smoothness* or *universality*, which ensures that if $\mathbf{c}$ is far from the q-ary lattice Λ generated by $\mathbf{A}$ (in particular if it is an encryption of 1), then given the projection key $\mathbf{p}$ (and $\mathbf{A}$ and $\mathbf{c}$), the prover cannot guess the hash value H with probability more than $1/2 + \mathsf{negl}(n)$. In [KV09], Katz and Vaikuntanathan argued universality for ciphertexts $\mathbf{c}$, for which every multiple of $\mathbf{c}$ is far from the lattice Λ. To be useful in their PAKE application, the decryption procedure of the encryption scheme therefore needs to be tweaked to try to decrypt not only the ciphertext itself but also all its multiples. In particular, their construction cannot work with super-polynomial moduli.

The question we focused on was whether universality holds without this tweak. In other words, is the condition that $j\mathbf{c}$ is far from Λ for all $j \neq 0$ truly necessary or is it is an artifact of the proof?

Harmonic analysis. The core of our work consists in using harmonic analysis to better understand the caveat of [KV09], namely that universality is only proven when all the multiples of the ciphertext are far from the lattice. For that, we extend the rounding function R to a q-periodic signal $\mathbb{R} \to \mathbb{R}$.

[3] In this section, I will use Word-independent SPHF instead of KV-SPHF to avoid the confusion between the lattice-based [KV09] and the initial word-independent SPHF in [KV11].



(a) $r(x) = \frac{1}{2} + \frac{1}{2}\cos\left(\frac{2\pi x}{q}\right)$

(b) $r(x) = r^\sharp(x) = 1 + \lfloor 2x/q \rceil \bmod 2$ (naive rounding)

(not to scale: T/q and $1/T$ are negligible)

(c) $r(x) = r^\flat(x)$ (used to smooth $r^\sharp$)

(not to scale: T/q and $1/T$ are negligible)

(d) $r(x) = (r^\sharp \odot r^\flat)(x)$

Figure 4.3: Probability that the rounding functions $R(x)$ output 1

We proceeded to a general analysis, which shows that universality holds for ciphertexts $\mathbf{c}$ such that its multiples $j\mathbf{c}$ are far away from the lattice Λ, for all non-zero integers j corresponding to non-zero real harmonics of the rounding signal R.

This unravels the causes of the caveat in [KV09]: the weight of the j-th harmonic of the naive rounding function $R\colon x \in \mathbb{Z}_q \mapsto \lfloor 2x/q \rceil \bmod 2$ (seen as a q-periodic signal, as in 4.3a) is as large as $\Theta(1/j)$ for odd integers j.

4.3.1 First solution (Universality, Approximate Correctness).

Having identified the source of the caveat, it becomes clear how to repair it: the rounding should be *randomized*, with a weight signal for which only the first harmonic is non-zero (in addition to the average), namely with a *pure cosine* weight:

$$\Pr[R(x) = 1] := \frac{1}{2} + \frac{1}{2}\cos\left(\frac{2\pi x}{q}\right) \quad .$$

This choice ensures universality as soon as just $1 \cdot \mathbf{c} = \mathbf{c}$ is far from the lattice Λ.

This solution nevertheless only provides approximate correctness (correctness holds with probability $3/4 + o(1)$), which is also problematic for some applications. This can be solved using correctness amplification via error-correcting codes, but at the price of preventing the resulting SPHF to be word-independent.

4.3.2 Second solution (Imperfect Universality, Statistical Correctness).

In our second instantiation, we therefore proceed to construct an almost-square rounding function, ($\odot$ denotes the convolution operator), which offers statistical correctness[4] and imperfect universality (namely the probability that a prover knowing only $\mathsf{hp} = \mathbf{p}$ can guess the hash value H is at most $1/3 + o(1)$). This instantiation requires a more subtle analysis, taking account of *destructive interferences*.

We can then amplify universality to get statistical universality (i.e., the above probability of guessing is at most $1/2 + \mathsf{negl}(n)$ as in our first solution) while keeping a statistical correctness. Contrary to the correctness amplification, this transformation preserves the independence of the projection key from the ciphertext. In particular, if the ciphertexts are from an IND-CPA scheme such as dual-Regev, then we get the first word-independent SPHF over a lattice-based language.

[4] More precisely, the probability of error is $\mathrm{poly}(n, \sigma)/q$, which is $\mathsf{negl}(n)$ for super-polynomial approximation factors q/σ.

We remark that our word-independent SPHF uses a *super-polynomial modulus* q, to get statistical correctness. It seems hard to construct such an SPHF for a polynomial modulus, as a word-independent SPHF for an IND-CPA encryption scheme directly yields a one-round key exchange (where each party sends a ciphertext of 0 and a projection key, and where the resulting session key is the xor of the two corresponding hash values) and we do not know of any lattice-based one-round key exchange using a polynomial modulus.

4.3.3 Mind the Gap

Even if we managed to solve the issue of building word-independent SPHF, and while doing so, we also allowed to use a simple encryption with a less tedious decryption process, there is still a major difference between *vanilla* SPHF, and these ones. The languages considered are now different in the smoothness and the correctness. In other words, the language considered for the correctness (and so honest user) is only a subset a subset of the language of words in the language for the smoothness.

Languages:

We consider a family of languages $(\mathscr{L}_{\mathsf{lpar},\mathsf{ltrap}})_{\mathsf{lpar},\mathsf{ltrap}}$ indexed by some *parameter* lpar and some *trapdoor* ltrap, together with a family of NP languages $(\widetilde{\mathscr{L}_{\mathsf{lpar}}})_{\mathsf{lpar}}$ indexed by some parameter lpar, with witness relation $\widetilde{\mathscr{R}}_{\mathsf{lpar}}$, such that:

$$\widetilde{\mathscr{L}_{\mathsf{lpar}}} = \{\chi \in \mathcal{X}_{\mathsf{lpar}} \mid \exists w,\ \widetilde{\mathscr{R}}_{\mathsf{lpar}}(\chi, w) = 1\} \ \subseteq\ \mathscr{L}_{\mathsf{lpar},\mathsf{ltrap}} \ \subseteq\ \mathcal{X}_{\mathsf{lpar}}\,,$$

where $(\mathcal{X}_{\mathsf{lpar}})_{\mathsf{lpar}}$ is a family of sets. The trapdoor ltrap and the parameter lpar are generated by a polynomial-time algorithm $\mathsf{Setup.lpar}$ which takes as input a unary representation of the security parameter n. We suppose that membership in $\mathcal{X}_{\mathsf{lpar}}$ and $\widetilde{\mathscr{R}}_{\mathsf{lpar}}$ can be checked in polynomial time given lpar and that membership in $\mathscr{L}_{\mathsf{lpar},\mathsf{ltrap}}$ can be checked in polynomial time given lpar and ltrap. The parameters lpar and ltrap are often omitted when they are clear from context.

We are mostly interested in languages of ciphertexts.

Languages of Ciphertexts Let $(\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a labeled encryption scheme. We define the following languages ($\mathsf{Setup.lpar} = \mathsf{KeyGen}$ and $(\mathsf{ltrap}, \mathsf{lpar}) = (\mathsf{dk}, \mathsf{ek})$):

$$\widetilde{\mathscr{L}} = \{(\mathsf{label}, C, M) \mid \exists \rho,\ C = \mathsf{Encrypt}(\mathsf{ek}, \mathsf{label}, M; \rho)\}\ ,$$
$$\mathscr{L} = \{(\mathsf{label}, C, M) \mid \mathsf{Decrypt}(\mathsf{dk}, \mathsf{label}, C) = M\}\ ,$$

where the witness relation $\widetilde{\mathscr{R}}$ is implicitly defined as: $\widetilde{\mathscr{R}}((\mathsf{label}, C, M), \rho) = 1$ if and only if $C = \mathsf{Encrypt}(\mathsf{ek}, \mathsf{label}, M; \rho)$.

Where in classical cryptography, we had $\widetilde{\mathscr{L}} = \mathscr{L}$, here with Euclidian Lattices, there is a gap. For most applications, one has to be careful during the proof [5], but in many cases the generic transformations still work once this discrepancy is taken into account.

This kind of problem seems to be inherent to the metric used. While studying new protocols on post-quantum cryptography, we had the idea to completely change the metric to have a different granularity and try to instantiate SPHF over Rank-Metric (code-based) cryptography.

4.4 And more

In addition to our lattice based constructions, we have also developed code based constructions for the NIST competition. This area of cryptography can be seen as behind in term of functionality when compared to lattice-based cryptography. However due to the existence of different based metrics, it allows a more fine-grained control over the noise.

With this in mind, we are building an hash proof system compatible with one of our contender for encryption in the NIST competition. We manage to show that not only we can build an hash proof system, but also thanks to the granularity of the underlying metric we manage to get rid of the gray area between the languages involved in the correctness, and the smoothness contrarily to the lattices results.

We now briefly sketch both the encryption, and the associated Hash Proof. Underlying ideas are close to the (naive) lattice schemes, except we have a way to test precisely if a word is well-formed, hence we get rid of corner cases living in the gap between correctness and smoothness.

[5] Intuitively, the simulator is going to try and process the adversary input, so he is going to use a trap to decrypt the input ciphertext, however this does not imply that the adversary sent a valid encryption

4.4.1 A Code-Based Encryption: RQC

RQC is a code-based IND-CCA2 encryption,we submitted to the NIST Post Quantum "Competition". Its security relies on the rank syndrome decoding problem without any additional assumption regarding the indistinguishability of the family of codes used [AMBD+18]. It is based on an IND-CPA construction denoted RQC.PKE (see figure 4.4) on top of which the HHK transformation [HHK17] is applied in order to obtain an IND-CCA2 cryptosystem.

RQC uses two types of codes: a Gabidulin code of generator matrix $\mathbf{G}$ denoted $\mathcal{C}$ and a random $[2n, n]$ quasi-cyclic code of parity-check matrix $(\mathbf{Id}\ \mathsf{rot}(\mathbf{h}))$ with $\mathbf{h}$ a random element of $\mathcal{V}$. One should note that the matrix $\mathbf{G}$ is public, hence the security of the scheme does not rely on the knowledge of the code $\mathcal{C}$ used.

- $\mathsf{Setup}(1^{\mathfrak{K}})$: Generates and outputs the global parameters $\mathsf{param} = (n, k, \delta, w, w_r, w_{\mathbf{e}}) \in \mathbb{N}^6$.
- $\mathsf{KeyGen}(\mathsf{param})$: Sets $\mathcal{R} = \mathcal{F}_{q^m}[X]/(X^n - 1)$. Samples $\mathbf{h} \xleftarrow{\$} \mathcal{R}$, the generator matrix $\mathbf{G} \in \mathcal{M}_{k,n}(\mathcal{F}_{q^m})$ of $\mathcal{C}$, $\mathsf{sk} = (\mathbf{x}, \mathbf{y}) \xleftarrow{\$} \mathcal{R}^2$ such that $\omega(\mathbf{x}) = \omega(\mathbf{y}) = w$, sets $\mathsf{pk} = (\mathbf{h}, \mathbf{s} = \mathbf{x} + \mathbf{h} \cdot \mathbf{y})$, and returns $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Encrypt}(\mathsf{pk}, \mathbf{m})$: Generates $\mathbf{r_3} \xleftarrow{\$} \mathcal{R}$, $\mathbf{r} = (\mathbf{r}_1, \mathbf{r}_2) \xleftarrow{\$} \mathcal{R}^2$ such that $\omega(\mathbf{r_3}) = w_{\mathbf{r_3}}$ and $\omega(\mathbf{r}_1) = \omega(\mathbf{r}_2) = w_r$, sets $\mathbf{c}_1 = \mathbf{r}_1 + \mathbf{h} \cdot \mathbf{r}_2$ and $\mathbf{c}_2 = \mathbf{mG} + \mathbf{s} \cdot \mathbf{r}_2 + \mathbf{r_3}$, returns $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2)$.
- $\mathsf{Decrypt}(\mathsf{sk}, \mathbf{c})$: Returns $\mathcal{C}.\mathsf{Decode}(\mathbf{c}_2 - \mathbf{y} \cdot \mathbf{c}_1)$.

Figure 4.4: Description of RQC.PKE

The correctness of RQC relies on the decoding capability of the Gabidulin code $\mathcal{C}$. While the indistinguishability relies on a decisional version of the Rank Syndrome Decoding Problem.

4.4.2 The associated Hash Proof

At a high level, one can view the scheme as a sort of a noisy ElGamal where $\mathbf{h}, \mathbf{s}$ play the role of the generators in the encryption key, and $\mathbf{y}$ the secret decryption key. With this in mind, we built an SPHF accordingly.

- $\mathsf{Setup}(1^{\mathfrak{K}})$: Given the security parameter $\mathfrak{K}$, generates the parameters param of the scheme namely $(n, m, k, q, w_r, w_\alpha, w_x) \in \mathbb{N}^7$, $\mathbf{G} \in \mathcal{M}_{k,n}(\mathcal{F}_{q^m})$ a generator matrix of a Gabidulin code $\mathcal{C}$, $\mathbf{h} \xleftarrow{\$} \mathcal{V}$ and $\mathbf{s} = \mathbf{x} + \mathbf{h} \cdot \mathbf{y}$ with $(\mathbf{x}, \mathbf{y}) \xleftarrow{\$} \mathcal{V}^2$ such that $\omega(\mathbf{x}) = \omega(\mathbf{y}) = w_x$.
- $\mathsf{HashKG}(\mathscr{L}_{\mathbf{m}}, \mathsf{param})$: Samples $(\boldsymbol{\alpha_1}, \boldsymbol{\alpha_2}, \boldsymbol{\alpha_3}, \boldsymbol{\alpha_4}) \xleftarrow{\$} \mathcal{V}^4$ such that $\forall i \in [\![1, 4]\!]$, $\mathsf{Supp}(\boldsymbol{\alpha_i}) = E_\alpha$ with $\dim(E_\alpha) = w_\alpha$. Returns $\mathsf{hk} = (\boldsymbol{\alpha_1}, \boldsymbol{\alpha_2}, \boldsymbol{\alpha_3}, \boldsymbol{\alpha_4})$.
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathscr{L}_{\mathbf{m}}, \mathsf{param}))$: Returns $\mathsf{hp} = \boldsymbol{\rho} = \mathbf{h} \cdot \boldsymbol{\alpha_1} + \mathbf{s} \cdot \boldsymbol{\alpha_2} + \boldsymbol{\alpha_3}$.
- $\mathsf{Hash}(\mathsf{hk}, (\mathscr{L}_{\mathbf{m}}, \mathsf{param}), \mathcal{W})$: Returns $\mathbf{v} = \boldsymbol{\alpha_1} \cdot \mathbf{c_1} + \boldsymbol{\alpha_2} \cdot (\mathbf{c_2} - \mathbf{mG}) + \boldsymbol{\alpha_4}$.
- $\mathsf{ProjHash}(\mathsf{hp}, (\mathscr{L}_{\mathbf{m}}, \mathsf{param}), \mathcal{W}, w)$: Sample $\mathbf{r_4} \xleftarrow{\$} \mathcal{V}$ such that $\mathsf{Supp}(\mathbf{r_4}) = E_r$ with $\dim(E_r) = w_r$. Returns $\mathbf{v}' = \boldsymbol{\rho} \cdot \mathbf{r_2} + \mathbf{r_4}$

Figure 4.5: A gapless code-based HPS

We then showed that for a correct choice of parameters, we can ensure that for well-formed ciphertexts in the language that v and v' were close enough, while for well-formed ciphertexts outside the language v was random under the decisional rank syndrome decoding.

Now, the main difference when compared with the lattice based approach is that we can check whether a ciphertext is well-formed. At least two natural ways come to mind:

- One can use a Stern-like Non-Interactive Zero-Knowledge proof. This allows every one to check publicly that the *prover* knows the randomness used in the encryption, while giving away the weight / support of the randomness hence showing it is not too big. On the plus side, this technique allows to check whether a word is a valid candidate before computing the hash value, however it is inefficient and requires to rely on the Random Oracle Model.
- Another solution that may be used when the prover sends the projected hash, is to compute the difference between the hash value, and the projected hash. Then using some LRPC decoding

technique the server can recover the randomness used, and checks whether it has the correct weight / support. The nice thing here is that this works in the standard model, however the SPHF can no longer be used with privacy in mind, and the verifier has to recover everything to be able to be convinced.

4.5 Expanding Languages

A major problem in cryptography is proving that we do not belong to a set. While proofs of membership have been known for years, proving a "non"-membership remains often complicated.

In [BCV15], we revisited [KZ09] and after showing a flaw in their design, we corrected their construction to handle Non-Interactive Zero-Knowledge Proofs of Non-Membership, but also to be construct Implicit Proofs of Non-Membership.

From a high level, the construction consists in building the proof of membership and seeing that it fails. There is however one caveat, one must also prove/check that the invalid *proof* was correctly computed (Otherwise it would be easy for anyone to pick a random value and say that it fails).

So the methodology leads to proving (and failing) that our words is in the language, and then proving (and succeeding) that this proof was correctly built.

Luckily as SPHF are often simple affine functions, it is easy to build another SPHF over them, checking that the projected Hash value was correctly computed.

To show that a word committed into $\mathbf{v}$ is not in a language described by $\mathscr{L}_p$, one ends up doing the following (cf Figure 4.6, page 25):

Prover $V, \mathbf{v}, w_v$		Verifier $\mathscr{L}_p, \mathscr{L}_u$
$(\mathsf{hp}_v, \mathsf{hk}_v) := (\mathsf{ProjKG}, \mathsf{HashKG})(\mathscr{L}_p))$		
$H_v := \mathsf{Hash}(\mathsf{hk}_v, \mathscr{L}_p, V)$	$\xrightarrow{\mathfrak{V} = (\mathsf{hp}_v, H_v, \mathbf{v})}$	$H'_v := \mathscr{L}_u \odot \mathsf{hp}_v \stackrel{?}{\neq} H_v$
	$\xleftarrow{\mathsf{hp}_u}$	$(\mathsf{hp}_u, \mathsf{hk}_u) := (\Gamma_u(\mathfrak{V}) \odot \alpha_{\mathbf{u}}, \alpha_{\mathbf{u}})$
$H'_u := (\mathbf{w_v}, \mathsf{hk}_{\mathbf{v}}) \odot \mathsf{hp}_u$	$\xrightarrow{H'_u}$	$H'_u \stackrel{?}{=} \Theta_u(\mathfrak{V}) \odot \mathsf{hk}_u.$

Figure 4.6: Generic SPHF-based proof of exclusion

Where the first SPHF is on the language described by $\mathscr{L}_p$ while the other one is based on the language of a correct computation between a hash value, a projection key and a ciphertext (as there is a dependency between those two terms it seems improbable to be able to do better in this case).

A more concrete example. In order to explain the previous formalism, let us now give a more concrete example, with a language described by an ElGamal encryption of a word U. The prover possesses a word V, the verifier the word U and publishes an ElGamal ciphertext of U: $\mathscr{L}_p = (h^s U, g^s)$. The prover encrypts his word V using ElGamal encryption scheme and proves to the verifier that V is not the plaintext encrypted in $\mathscr{L}_p$. Following the previous technique we can achieve a 3-round proof as described on Figure 4.7, page 25. The second SPHF is smooth if and only if $V = U$, this means that

Prover $V, h^r V, g^r, r$		Verifier $\mathscr{L}_p = (h^s U, g^s), U, s$
$(\mathsf{hp}_v, \mathsf{hk}_v) := (h^\lambda g^\mu, (\lambda, \mu))$	$\xrightarrow{\mathfrak{V}' = (\mathsf{hp}_v, h^r V, g^r)}$	$H'_v := \mathsf{hp}_v^s$
	$\xleftarrow{\mathsf{hp}_u, \mathscr{L}_p}$	$\mathsf{hp}_u := (h^\delta (h^s U/(h^r V))^\beta, g^\delta (g^s/g^r)^\beta,$ $\mathsf{hp}_v^\beta g^\gamma),$ $\mathsf{hk}_u := (\delta, \beta, \gamma)$
$H'_u := \mathsf{hp}_{u,1}^\lambda \mathsf{hp}_{u,2}^\mu \mathsf{hp}_{u,3}^r$ $H_v := (h^s U/V)^\lambda (g^s)^\mu$	$\xrightarrow{H'_u, H_v}$	$H'_v \stackrel{?}{\neq} H_v \wedge H'_u \stackrel{?}{=} \mathsf{hp}_v^\delta H_v^\beta (g^r)^\gamma.$

Figure 4.7: Tweaked ElGamal based SPHF proof of inequality

technically an adversary can break the soundness of the verification of the valid computation of hp_v, H_v when the word V is different from U. However in this case, the protocol should already return yes so he cannot gain anything from doing so. The proof requires overall 10 group elements: 2 for the initial commit of U, 2 for the one of V, 2 overall for hp_v, H_v and 4 for hp_u, H'_u.

We stress that, this construction differs from the generic approach in the sense that $\mathscr{L}_p$ instead of being known before the protocol like in the generic construction from Figure 4.6, page 25, can be set on the fly and postpone to the second flow.

Chapter 5

SPHF Friendly Commitment

Contents

As we managed to have SPHF tailored for the various *worlds* of cryptography, we were interested in having appropriate commitments. While there were several UC compatible commitment [CF01,CLOS02, DN02,CS03,Lin11,BCPV13], only a few of them [ACP09] were compatible with Smooth Projective Hash Functions with a very limited efficiency.

We proposed two commitments along those lines, first a generic constructions based on the Haralambiev commitment scheme [Har11] idea and a compatible CCA-2 encryption. Then, another one based on [FLM11] using our SP-SPHF methodology, that proves very compact, at the cost of using pairings.

This fits well with the classical approach that consists in using SPHF for the language of plaintext associated with an encryption.

5.1 Commitments.

We give here an informal overview to help the unfamiliar reader with the following. A *non-interactive labelled commitment scheme* $\mathcal{C}$ is defined by three algorithms:

- $\mathsf{Setup}(1^{\mathfrak{K}})$ takes as input the security parameter $\mathfrak{K}$ and outputs the global parameters, passed through the Common Reference String (CRS) to all other algorithms;
- $\mathsf{Com}^{\ell}(x)$ takes as input a label ℓ and a message x, and outputs a pair (C, δ), where C is the commitment of x for the label ℓ, and δ is the corresponding opening data. This is a probabilistic algorithm.
- $\mathsf{VerCom}^{\ell}(C, x, \delta)$ takes as input a commitment C, a label ℓ, a message x, and the opening data δ and outputs 1 (true) if δ is a valid opening data for C, x and ℓ. It always outputs 0 (false) on $x = \perp$.

The basic properties required for commitments are *correctness* (for all correctly generated CRS ρ, all commitments and opening data honestly generated pass the verification VerCom test), the *hiding property* (the commitment does not leak any information about the committed value) and the *binding property* (no adversary can open a commitment in two different ways).

A commitment scheme is said *equivocable* if it has a second setup $\mathsf{SetupComT}(1^{\mathfrak{K}})$ that additionally outputs a trapdoor τ, and two algorithms

- $\mathsf{SimCom}^{\ell}(\tau)$ that takes as input the trapdoor τ and a label ℓ and outputs a pair (C, eqk), where C is a commitment and eqk an equivocation key;
- $\mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, x)$ that takes as input a commitment C, a label ℓ, a message x, an equivocation key eqk, and outputs an opening data δ for C and ℓ on x.

such as the following properties are satisfied: *trapdoor correctness* (all simulated commitments can be opened on any message), *setup indistinguishability* (one cannot distinguish the CRS ρ generated by $\mathsf{SetupCom}$ from the one generated by $\mathsf{SetupComT}$) and *simulation indistinguishability* (one cannot distinguish a real commitment (generated by Com) from a fake commitment (generated by SCom), even with oracle access to fake commitments), denoting by SCom the algorithm that takes as input the trapdoor τ, a label ℓ and a message x and which outputs $(C, \delta) \xleftarrow{\$} \mathsf{SCom}^{\ell}(\tau, x)$, computed as $(C, \mathsf{eqk}) \xleftarrow{\$} \mathsf{SimCom}^{\ell}(\tau)$ and $\delta \leftarrow \mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, x)$.

A commitment scheme $\mathcal{C}$ is said *extractable* if it has a second setup $\mathsf{SetupComT}(1^{\mathfrak{K}})$ that additionally outputs a trapdoor τ, and a new algorithm

- $\mathsf{ExtCom}^{\ell}(\tau, C)$ which takes as input the trapdoor τ, a commitment C, and a label ℓ, and outputs the committed message x, or $\perp$ if the commitment is invalid.

such as the following properties are satisfied: *trapdoor correctness* (all commitments honestly generated can be correctly extracted: for all ℓ, x, if $(C, \delta) \xleftarrow{\$} \mathsf{Com}^{\ell}(x)$ then $\mathsf{ExtCom}^{\ell}(C, \tau) = x$), *setup indistinguishability* (as above) and *binding extractability* (one cannot fool the extractor, i.e. , produce a commitment and a valid opening data to an input x while the commitment does not extract to x).

5.2 Generic Commitment à la Haralambiev

In his doctoral dissertation, Haralambiev [Har11] proposed a commitment scheme that is both equivocable and extractable. The key was that the decommitment allowed the user to omit randomness, and so only *remember* the randomness corresponding to the value he wants to have committed. We showed in [ABB+13] how to adapt this commitment, to be compatible with SPHF, at the cost of using some pairings, we managed to propose a UC-Compatible SPHF-Friendly commitment that was only linear in the size of the value committed ([ACP09] was quadratic).

While revisiting this commitment in [BC15], we managed to propose a generalization compatible with the various worlds of cryptography.

We showed that one can build such a commitment, by combining a chameleon hash function (a hash function, with a trapdoor to generate collision) used to commit to every bit of the message, and a CCA-2 encryption of the randomness used in the chameleon hash. This construction no longer requires pairing, and we showed that those various tools can be instantiated under different assumption like DQR, DDH, or LWE.

5.2.1 Building Blocks.

We assume the existence of compatible CCA-encryption ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}$) and chameleon hash ($\mathsf{KeyGen}, \mathsf{VKeyGen}, \mathsf{CH}, \mathsf{Coll}, \mathsf{Valid}$), in the sense that is feasible to compute a CCA-encryption of the opening value of the chameleon hash. For example, a Pedersen Chameleon Hash is not compatible with Cramer Shoup encryption, as we would need to encrypt the randomness as a scalar, while the decryption algorithm only allows us to recover group elements.

In order for our commitment to accept an SPHF, we require the CCA-encryption to accept an SPHF on the language of valid ciphertexts. The precise language needed will depend on the way the chameleon hash is verified, but will be easily constructed by combining several simple languages as described in [BBC+13a].

We require the chameleon hash to be verifiable by the receiver. For the sake of brevity, we describe here the case where the chameleon hash is only verifiable by the server. In this case, we need a pre-flow, in which the server is assumed to execute the algorithm $\mathsf{VKeyGen}$ to generate a verification key and its trapdoor and send the verification key to the sender. This makes the commitment not completely non-interactive anymore but it should be noted that if the global protocol is not one-round, these values can be sent by the receiver during the first round of the protocol. In the case where the chameleon hash is publicly verifiable, one simply has to consider the keys vk and vtk empty, and ignore the pre-flow.

5.2.2 Generic Construction.

We now describe the different algorithms of our chameleon-hashed targeted commitment protocol CHCS from player P to Q (see Section 5.1 for the notations of the algorithms).

- **Setup and simulated setup algorithms:** $\mathsf{SetupComT}(1^{\mathfrak{K}})$ (the algorithm for setup with trapdoors) generates the various parameters param, for the setting of the SPHF-friendly labelled CCA-encryption scheme and the chameleon hash scheme. It then generates the corresponding keys and

trapdoors: $(\mathsf{ck}, \mathsf{tk})$ for the chameleon hash scheme and $(\mathsf{ek}, \mathsf{dk})$ for the encryption scheme.
For $\mathsf{SetupCom}(1^{\mathfrak{K}})$ (the algorithm for setup without trapdoors), the setting and the keys are generated the same way, but forgetting the way the keys were constructed (such as the scalars, in a DDH-based setting), thus without any trapdoor.
The algorithms both output the CRS $\rho = (\mathsf{ek}, \mathsf{ck}, \mathsf{param})$. In the first case, τ denotes the trapdoors $(\mathsf{dk}, \mathsf{tk})$.

- **Pre-flow (verification key generation algorithm):** player Q executes $\mathsf{VKeyGen}(\mathsf{ck})$ to generate the chameleon designated verification key vk and the trapdoor vtk and sends vk to the sender P.
- **Targeted commitment algorithm:** $\mathsf{Com}^{\ell}(\mathbf{M}; Q)$ from player P to player Q, for $\mathbf{M} = (M_i)_i \in \{0,1\}^m$ and a label ℓ, works as follows:
  - For $i \in [\![1, m]\!]$, it chooses r_{i,M_i} at random and computes $\mathsf{CH}(\mathsf{ck}, \mathsf{vk}, M_i; r_{i,M_i})$ to obtain the hash value a_i and the corresponding opening value d_{i,M_i}. It samples at random the values $r_{i,1-M_i}$ and $d_{i,1-M_i}$. We denote as $\mathbf{a} = (a_1, \ldots, a_m)$ the tuple of commitments and $\mathbf{d} = (d_{i,j})_{i,j}$.
  - For $i \in [\![1, m]\!]$ and $j = 0, 1$, it gets $\mathbf{b} = (b_{i,j})_{i,j} = \mathsf{2mEncrypt}^{\ell'}_{\mathsf{pk}}(\mathbf{d}; \mathbf{s})$, where $\mathbf{s}$ is taken at random and $\ell' = (\ell, \mathbf{a})$.

  The commitment is $C = (\mathbf{a}, \mathbf{b})$, and the opening information is the m-tuple $\delta = (s_{1,M_1}, \ldots, s_{m,M_m})$.
- **Verification algorithm:** $\mathsf{VerCom}^{\ell}(\mathsf{vtk}, C, \mathbf{M}, \delta)$ first checks the validity of the ciphertexts b_{i,M_i} with randomness s_{i,M_i}, then extracts d_{i,M_i} from b_{i,M_i} and s_{i,M_i}, and finally checks the chameleon hash a_i with opening value d_{i,M_i}, for $i \in [\![1, m]\!]$, via the algorithm $\mathsf{Valid}(\mathsf{ck}, \mathsf{vk}, M_i, a_i, d_{i,M_i}, \mathsf{vtk})$.
- **Simulated targeted commitment algorithm:** $\mathsf{SimCom}^{\ell}(\tau; Q)$ from the simulator to player Q, takes as input the equivocation trapdoor, namely tk, from $\tau = (\mathsf{dk}, \mathsf{tk})$, and outputs the commitment $C = (\mathbf{a}, \mathbf{b})$ and equivocation key $\mathsf{eqk} = \mathbf{s}$, where
  - For $i \in [\![1, m]\!]$, it chooses $r_{i,0}$ at random, computes $(a_i, d_{i,0}) = \mathsf{CH}(\mathsf{ck}, \mathsf{vk}, 0; r_{i,0})$, and uses the equivocation trapdoor tk to compute $r_{i,1}$ used to open the chameleon hash to 1 such that $\mathsf{CH}(\mathsf{ck}, \mathsf{vk}, 1; r_{i,1})$ is equal to $(a_i, d_{i,1})$. This leads to $\mathbf{a}$ and $\mathbf{d}$, making $d_{i,j}$ the opening value for $a_{i,j}$ for all $i \in [\![1, m]\!]$ and $j = 0, 1$.
  - $\mathbf{b}$ is built as above: $\mathbf{b} = (b_{i,j})_{i,j} = \mathsf{2mEncrypt}^{\ell'}_{\mathsf{pk}}(\mathbf{d}; \mathbf{s})$, where $\mathsf{eqk} = \mathbf{s}$ is taken at random and $\ell' = (\ell, \mathbf{a})$.
- **Equivocation algorithm:** $\mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, \mathbf{M})$ simply uses part of the equivocation key eqk (computed by the SimCom algorithm) to obtain the opening information $\delta = (s_{1,M_1}, \ldots, s_{m,M_m})$ in order to open to $\mathbf{M} = (M_i)_i$.
- **Extraction algorithm:** $\mathsf{ExtCom}^{\ell}(\tau, \mathsf{vtk}, C)$ takes as input the extraction trapdoor, namely the decryption key dk, from $\tau = (\mathsf{dk}, \mathsf{tk})$, the verification trapdoor vtk and a commitment $C = (\mathbf{a}, \mathbf{b})$. For $i \in [\![1, m]\!]$ and $j = 0, 1$, it first extracts the value $d_{i,j}$ from the ciphertext $b_{i,j}$, using the decryption key dk. Then, for $i \in [\![1, m]\!]$, it checks the chameleon hash a_i with opening values $d_{i,0}$ and $d_{i,1}$ with the help of the algorithm $\mathsf{Valid}(\mathsf{ck}, \mathsf{vk}, j, a_i, d_{i,j}, \mathsf{vtk})$ for $j = 0, 1$. If only one opening value $d_{i,j}$ satisfies the verification equality of the chameleon hash, then $j = M_i$. If this condition holds for each $i \in [\![1, m]\!]$, then the extraction algorithm outputs $(M_i)_i$. Otherwise (either if $\mathbf{b}$ could not be correctly decrypted, or there was an ambiguity while checking $\mathbf{a}$, with at least one chameleon hash a_i with two possible opening values $d_{i,0}$ and $d_{i,1}$), it outputs $\perp$.

Remark Given a publicly verifiable collision-resistant chameleon hash and a secure CCA-encryption accepting an SPHF on the language of valid ciphertexts, the above construction provides a commitment scheme which is SPHF-friendly for the languages:

$$\mathscr{L}_M = \left\{ (\ell, C) | \forall i \in [\![1, m]\!], \exists r_{i,M_i}, s_{i,M_i}, d_{i,M_i} \text{ such that } \begin{array}{l} \mathsf{mEncrypt}^{*,\ell}(\mathsf{pk}, (d_{i,M_i})_i; (s_{i,M_i})_i) = (b_{i,M_i})_i \\ \wedge\ \mathsf{CH}(\mathsf{ck}, \mathsf{vk}, M_i; r_{i,M_i}) = (a_i, d_{i,M_i}). \end{array} \right\}$$

5.3 Revisited FLM Commitment

It should be noted that the commitment used in [ACP09, ABB+13, BC15] has the major drawback of leaking the bit-length of the committed message (an upper-bound of it). While in application to Oblivious Transfer this is not a major problem, for PAKE this is a way more sensitive issue. The commitment proposed in [FLM11] is conceptually simpler, since the equivocation only needs to modify the witness, allowing the user to compute honestly its message in the commitment phase which lead to a more natural protocol execution.

5.3.1 k-MDDH Cramer-Shoup Encryption

[FLM11] heavily relies around Cramer Shoup encryption, so a first important step, is to supersede their DLin Cramer Shoup by a k-MDDH one. This was already done by [EHK+13], but we recall it here for sake of completeness.

- $\mathsf{Setup}(1^{\mathfrak{K}})$ generates a group $\mathbb{G}$ of order p, with an underlying matrix assumption using a base matrix $[\mathbf{A}] \in \mathbb{G}^{k+1\times k}$;
- $\mathsf{KeyGen}(\mathsf{param})$ generates $\mathsf{dk} = \mathbf{t}_1, \mathbf{t}_2, \mathbf{z} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$, and sets, $\mathbf{c} = \mathbf{t}_1\mathbf{A} \in \mathbb{Z}_p^k, \mathbf{d} = \mathbf{t}_2\mathbf{A} \in \mathbb{Z}_p^k, \mathbf{h} = \mathbf{z}\mathbf{A} \in \mathbb{Z}_p^k$. It also chooses a hash function $\mathfrak{H}_K$ in a collision-resistant hash family $\mathcal{H}$ (or simply a Universal One-Way Hash Function).
  The encryption key is $\mathsf{ek} = ([\mathbf{c}], [\mathbf{d}], [\mathbf{h}], [\mathbf{A}], \mathfrak{H}_K)$.
- $\mathsf{Encrypt}(\ell, \mathsf{ek}, [m]; \mathbf{r})$, for a message $M = [m] \in \mathbb{G}$ and random scalars $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^k$, the ciphertext is $C = (\mathbf{u} = [\mathbf{A}\mathbf{r}]), e = [\mathbf{h}\mathbf{r} + m], v = [(\mathbf{c} + \mathbf{d} \odot \xi)\mathbf{r}]_1$, where v is computed afterwards with $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$.
- $\mathsf{Decrypt}(\ell, \mathsf{dk}, C)$: one first computes $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$ and checks whether v is consistent with $\mathbf{t}_1, \mathbf{t}_2$. If it is, one computes $M = [e - (\mathbf{u}\mathbf{z})]$ and outputs M. Otherwise, one outputs $\perp$.

Theorem 5.3.1 *The k-MDDH Cramer-Shoup Encryption is IND-CCA 2 under k-MDDH assumption and the collision resistance (universal one-wayness) of the Hash Family.*

Structure-Preserving Smooth Projective Hash Function For ease of readability we set $\mathbf{B} = \begin{bmatrix} \begin{pmatrix} h \\ \mathbf{A} \\ \mathbf{c} \end{pmatrix} \end{bmatrix}$ and $\mathbf{D} = \begin{bmatrix} \begin{pmatrix} 0 \\ \vdots \\ \mathbf{d} \end{pmatrix} \end{bmatrix}$, and write $C' = [\mathbf{B}\mathbf{r} + \xi\mathbf{D}\mathbf{r}]_1$ the ciphertext without the message M.

- $\mathsf{HashKG}(\mathscr{L}, \mathsf{param})$, chooses $\Lambda \xleftarrow{\$} \mathbb{Z}_p^{(k+2)\times 1}$, $\lambda \xleftarrow{\$} \mathbb{Z}_p$ and sets $\mathsf{hk}_1 = \Lambda, \mathsf{hk}_2 = \begin{pmatrix} \lambda \\ \mathbf{0} \\ \Lambda_{k+2} \end{pmatrix}$;
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathscr{L}, \mathsf{param}), W)$, outputs $\mathsf{hp}_1 = \mathsf{hk}_1^\top \mathbf{B}, \mathsf{hp}_2 = \mathsf{hk}_2^\top \begin{pmatrix} h \\ \mathbf{0} \\ d \end{pmatrix}$;
- $\mathsf{Hash}(\mathsf{hk}, (\mathscr{L}, \mathsf{param}), W)$, outputs a hash value $H = [(\mathsf{hk}_1 + \xi\mathsf{hk}_2)^\top C']_T$;
- $\mathsf{ProjHash}(\mathsf{hp}, (\mathscr{L}, \mathsf{param}), W, w)$, outputs the value $H' = [(\mathsf{hp}_1 + \xi\mathsf{hp}_2)\mathbf{r}]_T$.

The Smoothness comes inherently from the fact that we have $2k + 2$ unknowns in hk while hp gives at most $2k$ equations. Hence an adversary has a negligible chance to find the real values.

5.3.2 A Universally Composable Commitment with Adaptive Security Based on MDDH

We first show how to simply generalize FLM's commitment [FLM11] from DLin to $\mathbf{k}$-MDDH.

At Asiacrypt 2011, Fischlin, Libert and Manulis presented a universally composable commitment [FLM11] with adaptive security based on the Decision Linear assumption [BBS04]. We show here how to generalize their scheme to the Matrix Decisional Diffie-Hellman assumption from [EHK+13]. Note that sid denotes the session identifier and cid the commitment identifier and that the combination $(\mathsf{sid}, \mathsf{cid})$ is globally unique, as in [HMQ04, FLM11].

Compared to the original version of the commitment, we split the proof $\pi_{val-enc}$ into its two parts: the NIZK proof denoted here as $[\mathbf{\Pi}]_1$ is still revealed during the opening algorithm, while the Groth-Sahai commitment $[\mathbf{R}]_2$ of the randomness $\mathbf{r}$ of the Cramer-Shoup encryption is sent during the commitment phase. Furthermore, since the hash value in the Cramer Shoup encryption is used to link the commitment with the session, we include this value $[\mathbf{R}]_2$ to the label, in order to ensure that this extra commitment information given with the ciphertext is the original one. We refer the reader to the original security proof in [FLM11, Theorem 1], which remains exactly the same, since this additional commitment provides no information (either computationally or perfectly, depending on the CRS), and since the commitment $[\mathbf{R}]_2$ is not modified in the equivocation step (only the value $[\mathbf{\Pi}]_1$ is changed).

- **CRS Generation:** algorithm $\mathsf{SetupCom}(1^{\mathfrak{K}})$ chooses a bilinear asymmetric group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ of order $p > 2^{\mathfrak{K}}$, and a set of generators $[\mathbf{A}]_1$ corresponding to the underlying matrix assumption. As explained in [EHK+13], following their notations, one can define a Groth-Sahai CRS by picking $\mathbf{w} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$, and setting $[\mathbf{U}]_2 = [\mathbf{B}||\mathbf{B}\mathbf{w}]_2$ for a hiding CRS, and $[\mathbf{B}||\mathbf{B}\mathbf{w} + (0||z)^\top]_2$ otherwise, where $[\mathbf{B}]_2$ is an k-MDDH basis, and $\mathbf{w}, z$ are the elements defining the challenge vector.

For the Cramer-Shoup like CCA-2 encryption, one additionally picks $\mathbf{t}_1, \mathbf{t}_2, \mathbf{z} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$, and a Universal One-Way Hash Function $\mathcal{H}$ and sets $[\mathbf{h}]_1 = [\mathbf{z} \cdot \mathbf{A}]_1$, $[\mathbf{c}]_1 = [\mathbf{t}_1\mathbf{A}]_1, [\mathbf{d}]_1 = [\mathbf{t}_2\mathbf{A}]_1$.
The CRS consists of $\mathsf{crs} = (\mathfrak{K}, p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, [\mathbf{A}]_1 \in \mathbb{G}_1^{k\times k+1}, [\mathbf{U}]_2, [\mathbf{h}]_1 \in \mathbb{G}_1^k, [\mathbf{c}]_1 \in \mathbb{G}_1^k, [\mathbf{d}]_1 \in \mathbb{G}_1^k, \mathcal{H})$.

- **Commitment algorithm:** $\mathsf{Com}(\mathsf{crs}, M, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$, to commit to message $M \in \mathbb{G}_1$ for party P_j, party P_i conducts the following steps:
  - It chooses random exponents $\mathbf{r}$ in $\mathbb{Z}_p^k$ and commits to $\mathbf{r}$ in $[\mathbf{R}]_2$ with randomness $\rho \xleftarrow{\$} \mathbb{Z}_p^{k\times k+1}$, setting $[\mathbf{R}]_2 = [\mathbf{U}\rho + \iota_2(\mathbf{r})]_2 \in \mathbb{G}_2^{k\times k+1}$. It also computes a Cramer-Shoup encryption $\psi_{CS} = [\mathbf{C}]_1$ of $M \in \mathbb{G}_1$ under the label $\ell = P_i \| \mathsf{sid} \| \mathsf{cid}$ and the public key pk:
    $$[\mathbf{C}]_1 = [\mathbf{A}\mathbf{r} || \mathbf{h}\mathbf{r} + M || (\mathbf{c} + \mathbf{d} \odot \mathcal{H}(\ell || \mathbf{C}_1 || \mathbf{C}_2 || \mathbf{R}))\mathbf{r}]_1 = [\mathbf{C}_1 || \mathbf{C}_2 || \mathbf{C}_3]_1$$
    For simplicity we write $\ell' = \ell || [\mathbf{C}_1]_1 || [\mathbf{C}_2]_1 || [\mathbf{R}]_2$.
  - It generates a NIZK proof $D_M = [\mathbf{\Pi}]_1$ that ψ_{CS} is indeed a valid encryption of $M \in \mathbb{G}_1$ for the committed $\mathbf{r}$ in $[\mathbf{R}]_2$. This requires to prove that these exponents satisfy the multi-exponentiation equations:
    $$[\mathbf{C}_1]_1 = [\mathbf{A}\mathbf{r}]_1, [\mathbf{C}_2 - M]_1 = [\mathbf{h}\mathbf{r}]_1, [\mathbf{C}_3 = (\mathbf{c} + \mathbf{d} \odot \mathcal{H}(\ell'))\mathbf{r}]_1$$
    The associated proof is then $[\mathbf{\Pi}]_1 = [\rho^\top(\mathbf{A} || \mathbf{h} || \mathbf{c} + \mathbf{d} \odot \mathcal{H}(\ell'))]_1$.
  - P_i erases r after the generation of $[\mathbf{R}]_2$ and $[\mathbf{\Pi}]_1$ but retains $D_M = [\mathbf{\Pi}]_1$.

  The commitment is $([\mathbf{C}]_1, [\mathbf{R}]_2)$.
- **Verification algorithm:** the algorithm $\mathsf{VerCom}(\mathsf{crs}, M, D_M, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$ checks the consistency of the proof $\pi_{val-enc}$ with respect to $[\mathbf{C}]_1$ and $[\mathbf{R}]_2$. and ignores the opening if the verification fails.
- **Opening algorithm:** $\mathsf{OpenCom}(\mathsf{crs}, M, D_M, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$ reveals M and $D_M = [\Pi]_1$ to P_j.

One can easily see that $[\mathbf{C}_3]_1$ is the projective hash computation of a 2-universal hash proof on the language "$[\mathbf{C}_1]_1$ in the span of $\mathbf{A}$", with $[\mathbf{C}_2]_1$ being an additional term that uses the same witness to mask the committed message, so that $[\mathbf{C}]_1$ is a proper generalization of the Cramer-Shoup CCA-2 encryption. Details on the k-MDDH Groth-Sahai proofs are given in [BC16] but the generalization is natural.

It is thus easy to see that this commitment is indeed a generalization of the FLM non-interactive UC commitment with adaptive corruption under reliable erasures.

5.3.3 Associated Structure-Preserving Smooth Projective Hash Function

We now want to supersede the verification equation of the commitment by a smooth projective hash function providing implicit decommitment, simply using the proof as a witness. We consider the language of the valid encryptions of M using a random r which is committed into $[\mathbf{R}]_2$:

$$\mathscr{L}_M = \{[\mathbf{C}]_1 \mid \exists r \exists \rho \text{ such that } [\mathbf{R}]_2 = [\mathbf{U}\rho + \iota_2(\mathbf{r})]_2 \\ \text{and } [\mathbf{C}]_1 = [\mathbf{A}\mathbf{r} || \mathbf{h}\mathbf{r} + M || (\mathbf{c} + \mathbf{d} \odot \mathcal{H}(\ell || \mathbf{C}_1 || \mathbf{C}_2 || \mathbf{R}))\mathbf{r}]_1\}$$

The verifier picks a random $\mathsf{hk} = \alpha \xleftarrow{\$} \mathbb{Z}_p^{k+3\times k+1}$ and sets $\mathsf{hp} = [\alpha \odot \mathbf{U}]_2$.
On one side, the verifier then computes:

$$\mathsf{Hash}(\mathsf{hk}, ([\mathbf{C}]_1, [\mathbf{R}]_2)) = [\alpha \odot ((\mathbf{C}_1 || \mathbf{C}_2 - M || \mathbf{C}_3) - (\mathbf{A} || \mathbf{h} || \mathbf{c} + \mathbf{d} \odot \mathcal{H}(\ell')) \cdot \mathbf{R})]_T$$

While the prover computes $\mathsf{ProjHash}(\mathsf{hp}, \mathbf{\Pi}) = [\mathbf{\Pi} \cdot \mathsf{hp}]_T$.

- *Correctness*: comes directly from the previous equations.
- *Smoothness*: on a binding CRS, $[\mathbf{U}]_2$'s last column is in the span of the k first (which are simply $[\mathbf{B}]_2$), hence as $\mathsf{hk} \in \mathbb{Z}_p^{k+1}$, the k equations given in hp are not enough to determine its value and so it is still perfectly hidden from an information theoretic point of view.

Efficiency. The rough size of a projection key is $k \times (k+3)$ (number of elements in each proof times number of proofs). It should be noted that if we do not need a KV-SPHF (in the case of oblivious transfer for example), instead of repeating the projection key $k+3$ times (in order to verify each component of the Cramer-Shoup), one can generate a value $\varepsilon \xleftarrow{\$} \mathbb{Z}_p$, an hp for a single equation, and say that for the other component, one simply uses $\mathsf{hp}^{\varepsilon^i}$.

$\star$

Part II

Using HPS in Constructions

Chapter 6

Symmetric Constructions (LAKE)

Contents

Beyond the original Chosen-Ciphertext secure encryption scheme of Cramer and Shoup [CS98], Smooth Projective Hash Functions have given rise to generalized classes of Authenticated Key Exchange (Password-based, Language-based, ...) [GL06, ACP09, KV11, BBC+13a]. They also have been used in Oblivious Transfer [Kal05, HK12] One-Time Relatively-Sound Non-Interactive Zero-Knowledge Arguments [JR12], and Zero-Knowledge Arguments [BBC+13b].

In this chapter, we are going to focus on symmetric constructions where we expect some form of mutual authentication.

6.1 Language Authenticated Key Exchange

In [BBC+13a], we introduced a new notion, under the name of Language Authenticated Key Exchange which was supposed to encompass every possible authenticated two-party key exchange. We assume the existence of languages $\mathscr{L}_i, \mathscr{L}_j$ and wanted a protocol that leads to a shared key between users $\mathcal{U}_i$ and $\mathcal{U}_j$ if and if $\mathcal{U}_i$ possesses a word $\mathcal{W}_i \in \mathscr{L}_i$ and $\mathcal{U}_j$ possesses a word $\mathcal{W}_j \in \mathscr{L}_j$.

This notion naturally encompasses the concept of Password-Authenticated Key Exchange [BM92] (where $\mathscr{L}_i = \mathscr{L}_j = \{\mathsf{pw}\}$), Secret Handshakes [BDS+03] (where $\mathscr{L}_i = \mathscr{L}_j = \{(m, \sigma) | \mathsf{Valid}(\mathsf{vk}, \sigma, m)\}$) or even the notion of Credential Authenticated Key Exchange [CCGS10] where Credentials have to be *compatible*.

In order to define the security of this primitive, we use the UC framework and an appropriate definition for languages that permits to dissociate the public part of the policy, the private common information the users want to check and the (possibly independent) secret values each user owns that assess the membership to the languages. We provide an ideal functionality for LAKE and give efficient realizations of the new primitive (for a large family of languages) secure under classical mild assumptions, in the standard model (with a common reference string – CRS), with static corruptions.

Language definition.

In [ACP09], Abdalla *et al.* already formalized languages to be considered for SPHF. But, in the following, we will use a more simple formalism, which is nevertheless more general: we consider any efficiently computable binary relation $\mathcal{R} : \{0,1\}^* \times \mathcal{P} \times \mathcal{S} \to \{0,1\}$, where the additional parameters $\mathsf{pub} \in \{0,1\}^*$ and $\mathsf{priv} \in \mathcal{P}$ define a language $\mathscr{L}_{\mathcal{R}}(\mathsf{pub}, \mathsf{priv}) \subseteq \mathcal{S}$ of the words W such that $\mathcal{R}(\mathsf{pub}, \mathsf{priv}, W) = 1$:

- pub are public parameters;
- priv are private parameters the two players have in mind, and they should think to the same values: they will be committed to, but never revealed;
- W is the word the sender claims to know in the language: it will be committed to, but never revealed.

Our LAKE primitive, specific to two relations $\mathcal{R}_a$ and $\mathcal{R}_b$, will allow two users, Alice and Bob, owning a word $W_a \in \mathscr{L}_{\mathcal{R}_a}(\mathsf{pub}, \mathsf{priv}_a)$ and $W_b \in \mathscr{L}_{\mathcal{R}_b}(\mathsf{pub}, \mathsf{priv}_b)$ respectively, to agree on a session key under some specific conditions: they first both agree on the public parameter pub, but Bob will think about priv'_a for his expected value of priv_a, Alice will do the same with priv'_b for priv_b; eventually, if $\mathsf{priv}'_a = \mathsf{priv}_a$ and $\mathsf{priv}'_b = \mathsf{priv}_b$, and if they both know words in the languages, then the key agreement will succeed. In case of failure, no information should leak about the reason of failure, except the inputs did not satisfy the relations $\mathcal{R}_a$ or $\mathcal{R}_b$, or the languages were not consistent.

We stress that each LAKE protocol will be specific to a pair of relations $(\mathcal{R}_a, \mathcal{R}_b)$ describing the way Alice and Bob will authenticate to each other. This pair of relations $(\mathcal{R}_a, \mathcal{R}_b)$ specifies the sets $\mathcal{P}_a$, $\mathcal{P}_b$ and $\mathcal{S}_a$, $\mathcal{S}_b$ (to which the private parameters and the words should respectively belong). Therefore, the formats of priv_a, priv_b and W_a and W_b are known in advance, but not their values. When $\mathcal{R}_a$ and $\mathcal{R}_b$ are clearly defined from the context (e.g., PAKE), we omit them in the notations. For example, these relations can formalize:

- Password-based authentication: The language is defined by $\mathcal{R}(\mathsf{pub}, \mathsf{priv}, W) = 1 \Leftrightarrow W = \mathsf{priv}$, and thus $\mathsf{pub} = \emptyset$. The classical setting of PAKE requires the players A and B to use the same password W, and thus we should have $\mathsf{priv}_a = \mathsf{priv}'_b = \mathsf{priv}_b = \mathsf{priv}'_a = W_a = W_b$;
- Signature-based authentication: $\mathcal{R}(\mathsf{pub}, \mathsf{priv}, W) = 1 \Leftrightarrow \mathsf{Verif}(\mathsf{pub}_1, \mathsf{pub}_2, W) = 1$, where $\mathsf{pub} = (\mathsf{pub}_1 = \mathsf{vk}, \mathsf{pub}_2 = M)$ and $\mathsf{priv} = \emptyset$. The word W is thus a signature of M valid under vk, both specified in pub;
- Credential-based authentication: we can consider any mix for vk and M in pub or priv, and even in W, for which the relation $\mathcal{R}$ verifies the validity of the signature. When M and vk are in priv or W, we achieve *affiliation-hiding* property.

In the two last cases, the parameter pub can thus consist of a message on which the user is expected to know a signature valid under vk: either the user knows the signing key and can generate the signature on the fly to run the protocol, or the user has been given signatures on some messages (credentials). As a consequence, we just assume that, after having publicly agreed on a common pub, the two players have valid words in the appropriate languages. The way they have obtained these words does not matter.

6.1.1 The Ideal Functionality

We generalize the Password-Authenticated Key Exchange functionality $\mathcal{F}_{\text{PAKE}}$ (first provided in [CHK+05]) to more complex languages: the players agree on a common secret key if and only if they own words that lie in the languages the partners have in mind. As before in this paper, the languages are formalized by an efficiently computable binary relation $\mathcal{R} : \{0,1\}^* \times \mathcal{P} \times \mathcal{S} \to \{0,1\}$ which defines the words $W \in \mathcal{S}$ that are in the language $\mathscr{L}_{\mathcal{R}}(\mathsf{pub}, \mathsf{priv})$, according to the public part $\mathsf{pub} \in \{0,1\}^*$ and the private part $\mathsf{priv} \in \mathcal{P}$. More precisely, after an agreement on pub between P_i and P_j (modeled here by the use of the split functionality, see below), player P_i uses a word W_i belonging to $\mathscr{L}_i = \mathscr{L}_{\mathcal{R}_i}(\mathsf{pub}, \mathsf{priv}_i)$ and it expects its partner P_j to use a word W_j belonging to the language $\mathscr{L}'_j = \mathscr{L}_{\mathcal{R}_j}(\mathsf{pub}, \mathsf{priv}'_j)$. We assume relations $\mathcal{R}_i$ and $\mathcal{R}_j$ to be specified by the kind of protocol we study (PAKE, Verifier-based PAKE, secret handshakes, ...) and so the languages are defined by the additional parameters pub, priv_i and priv_j only: they both agree on the public part pub, to be possibly parsed in a different way by each player for each language according to the relations, player P_i owns $W_i \in \mathscr{L}_i = \mathscr{L}(\mathsf{pub}, \mathsf{priv}_i) \subseteq \mathcal{S}_i$, for $\mathsf{priv}_i \in \mathcal{P}_i$, and expects player P_j to use the language $L'_j = \mathscr{L}(\mathsf{pub}, \mathsf{priv}'_j) \subseteq \mathcal{S}_j$, for $\mathsf{priv}'_j \in \mathcal{P}_j$. Symmetrically, player P_j owns $W_j \in \mathscr{L}_j = \mathscr{L}(\mathsf{pub}, \mathsf{priv}_j) \subseteq \mathcal{S}_j$ and expects player P_i to use the language $L'_i = \mathscr{L}(\mathsf{pub}, \mathsf{priv}'_i) \subseteq \mathcal{S}_i$. The subsets $\mathcal{S}_i, \mathcal{S}_j$ and $\mathcal{P}_i, \mathcal{P}_j$ are assumed public and determined by $\mathcal{R}_i$ and $\mathcal{R}_j$, and thus by the kind of protocol, and known in advance.

Note however that the respective languages do not need to be the same or to use similar relations: authentication means could be totally different for the 2 players. The key exchange should succeed if and

The functionality $\mathcal{F}_{\text{LAKE}}$ is parametrized by a security parameter k and a public parameter pub for the languages. It interacts with an adversary $\mathcal{S}$ and a set of parties $P_1,\ldots,P_n$ via the following queries:

- New Session: Upon receiving a query $(\mathit{NewSession} : \mathsf{sid}, P_i, P_j, W_i, \mathscr{L}_i = \mathscr{L}(\mathsf{pub}, \mathsf{priv}_i), \mathscr{L}'_j = \mathscr{L}(\mathsf{pub}, \mathsf{priv}'_j))$ from P_i,
  - If this is the first *NewSession*-query with identifier sid, record the tuple $(P_i, P_j, W_i, \mathscr{L}_i, \mathscr{L}'_j, \mathsf{initiator})$. Send $(\mathit{NewSession}; \mathsf{sid}, P_i, P_j, \mathsf{pub}, \mathsf{initiator})$ to $\mathcal{S}$ and P_j.
  - If this is the second *NewSession*-query with identifier sid and there is a record $(P_j, P_i, W_j, \mathscr{L}_j, \mathscr{L}'_i, \mathsf{initiator})$, record the tuple $(P_j, P_i, W_j, \mathscr{L}_j, \mathscr{L}'_i, \mathsf{initiator}, W_i, \mathscr{L}_i, \mathscr{L}'_j, \mathsf{receiver})$. Send $(\mathit{NewSession}; \mathsf{sid}, P_i, P_j, \mathsf{pub}, \mathsf{receiver})$ to $\mathcal{S}$ and P_j.
- Key Computation: Upon receiving a query $(\mathit{NewKey} : \mathsf{sid})$ from $\mathcal{S}$, if there is a record of the form $(P_i, P_j, W_i, \mathscr{L}_i, \mathscr{L}'_j, \mathsf{initiator}, W_j, \mathscr{L}_j, \mathscr{L}'_i, \mathsf{receiver})$ and this is the first *NewKey*-query for session sid, then
  - If $(\mathscr{L}'_i = \mathscr{L}_i$ and $W_i \in \mathscr{L}_i)$ and $(\mathscr{L}'_j = \mathscr{L}_j$ and $W_j \in \mathscr{L}_j)$, then pick a random key sk of length k and store $(\mathsf{sid}, \mathsf{sk})$. If one player is corrupted, send $(\mathsf{sid}, \mathsf{success})$ to the adversary.
  - Else, store $(\mathsf{sid}, \perp)$, and send $(\mathsf{sid}, \mathsf{fail})$ to the adversary if one player is corrupted.
- Key Delivery: Upon receiving a query $(\mathit{SendKey} : \mathsf{sid}, P_i, \mathsf{sk})$ from $\mathcal{S}$, then
  - if there is a record of the form $(\mathsf{sid}, \mathsf{sk}')$, then, if both players are uncorrupted, output $(\mathsf{sid}, \mathsf{sk}')$ to P_i. Otherwise, output $(\mathsf{sid}, \mathsf{sk})$ to P_i.
  - if there is a record of the form $(\mathsf{sid}, \perp)$, then pick a random key sk' of length k and output $(\mathsf{sid}, \mathsf{sk}')$ to P_i.

Figure 6.1: Ideal Functionality $\mathcal{F}_{\text{LAKE}}$

only if the two following pairs of equations hold: $(\mathscr{L}'_i = \mathscr{L}_i$ and $W_i \in \mathscr{L}_i)$ and $(\mathscr{L}'_j = \mathscr{L}_j$ and $W_j \in \mathscr{L}_j)$.

Description.

In the initial $\mathcal{F}_{\text{PAKE}}$ functionality [CHK+05], the adversary was given access to a $\mathsf{TestPwd}$-query, which modeled the on-line dictionary attack. But it is known since [BCL+05] that it is equivalent to use the split functionality model [BCL+05], generate the *NewSession*-queries corresponding to the corrupted players and tell the adversary (on behalf of the corrupted player) whether the protocol should succeed or not. Both methods enable the adversary to try a credential for a player (on-line dictionary attack). The second method (that we use here) implies allowing $\mathcal{S}$ to ask *NewSession*-queries on behalf of the corrupted player, and letting it to be aware of the success or failure of the protocol in this case: the adversary learns this information only when it plays on behalf of a player (corruption or impersonation attempt). This is any way an information it would learn at the end of the protocol. We insist that third parties will not learn whether the protocol succeeded or not, as required for secret handshakes. To this aim, the *NewKey*-query informs in this case the adversary whether the credentials are consistent with the languages or not. In addition, the split functionality model guarantees from the beginning which player is honest and which one is controlled by the adversary. This finally allows us to get rid of the $\mathsf{TestPwd}$-query. The $\mathcal{F}_{\text{LAKE}}$ functionality is presented in Figure 6.1, page 35.

6.1.2 Generic Construction

Using smooth projective hash functions on commitments, one can generically define a LAKE protocol as done in [ACP09]. The basic idea is to make the player commit to their private information (for the expected languages and the owned words), and eventually the smooth projective hash functions will be used to make implicit validity checks of the global relation as described in Figure 6.2, page 36. For simplicity, we assume every language can be checked using a KV-SPHF, [BBC+13a] gives a more detailed version otherwise, but it requires additional rounds.

In the rest of this chapter, we are going to detail more specific instantiations of this framework to achieve various protocols. Interestingly, everytime we managed to obtain either the most efficient protocol to date, or at least close at the given level of security.

Execution between P_i and P_j, with session identifier sid.
We denote by $\ell_i = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{pub}, \mathsf{VK}_i, \mathsf{VK}_j)$ and by $\ell_j = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{pub}, \mathsf{VK}_j, \mathsf{VK}_i)$, where pub is the combination of the contributions of the two players. The initiator now uses a word W_i in the language $\mathscr{L}(\mathsf{pub}, \mathsf{priv}_i)$, and the receiver uses a word W_j in the language $\mathscr{L}(\mathsf{pub}, \mathsf{priv}_j)$ (*). We assume commitments and associated smooth projective hash functions exist for these languages.

Exchange Each user P_i (with random tape ω_i) generates a commitment on $(\mathsf{priv}_i, \mathsf{priv}'_j, W_i)$ in $\mathcal{C}_i$, where W_i has been randomized in the language, under the label ℓ_i. It also generated $\mathsf{hp}_j, \mathsf{hk}_j$ for the language $\mathscr{L}_j$. And then sends $\mathcal{C}_i, \mathsf{hp}_j$ to P_j;

Hashing Each P_i, computes $K_i = H_i \oplus H'_j$ as follows:

$$\begin{aligned} H_i &= \mathsf{Hash}(\mathsf{hk}_j, \{(\mathsf{priv}'_j, \mathsf{priv}_i)\} \times \mathscr{L}(\mathsf{pub}, \mathsf{priv}'_j), \ell_j, \mathsf{Com}_j) \\ H'_j &= \mathsf{ProjHash}(\mathsf{hp}_i, \{(\mathsf{priv}_i, \mathsf{priv}'_j)\} \times \mathscr{L}(\mathsf{pub}, \mathsf{priv}_i), \ell_i, \mathsf{Com}_i, \omega_i) \end{aligned}$$

(*) As explained in Section 6.1, page 34, recall that the languages considered depend on two possibly different relations, namely $\mathscr{L}_i = \mathscr{L}_{\mathcal{R}_i}(\mathsf{pub}, \mathsf{priv}_i)$ and $\mathscr{L}_j = \mathscr{L}_{\mathcal{R}_j}(\mathsf{pub}, \mathsf{priv}_j)$, but we omit them for the sake of clarity. We assume they are both self-randomizable.

Figure 6.2: Language-based Authenticated Key Exchange from a Smooth Projective Hash Function on Commitments

6.2 Password-Authenticated Key Exchange

Password-Authenticated Key Exchange (PAKE) protocols were proposed in 1992 by Bellovin and Merritt [BM92] where authentication is done using a simple password, possibly drawn from a small entropy space subject to exhaustive search. Since then, many schemes have been proposed and studied. SPHF have been extensively used, starting with the work of Gennaro and Lindell [GL03] which generalized an earlier construction by Katz, Ostrovsky, and Yung [KOY01], and followed by several other works [CHK+05, ACP09]. More recently, a variant proposed by Katz and Vaikuntanathan even allowed the construction of one-round PAKE schemes [KV11, BBC+13b]. [1]

The first ideal functionality for PAKE protocols in the UC framework [Can01, CK02] was proposed by Canetti *et al.* [CHK+05], who showed how a simple variant of the Gennaro-Lindell methodology [GL03] could lead to a secure protocol. Though quite efficient, their protocol was not known to be secure against adaptive adversaries, that are capable of corrupting players at any time, and learn their internal states. The first ones to propose an adaptively secure PAKE in the UC framework were Barak *et al.* [BCL+05] using general techniques from multi-party computation. Though conceptually simple, their solution results in quite inefficient schemes.

Recent adaptively secure PAKE were proposed by Abdalla *et al.* [ACP09, ABB+13], following the Gennaro-Lindell methodology with variation of the Canetti-Fischlin commitment [CF01]. However their communication size is growing in the size of the passwords, which is leaking information about an upper-bound on the password used in each exchange.

6.2.1 Ideal Functionality

We present the PAKE ideal functionality $\mathcal{F}_{pwKE}$ on Figure 6.3, page 37. It was described in [CHK+05].

The main idea behind this functionality is as follows: If neither party is corrupted and the adversary does not attempt any password guess, then the two players both end up with either the same uniformly-distributed session key if the passwords are the same, otherwise they have uniformly-distributed independent session keys. In addition, the adversary does not learn whether the interaction was successful. However, if one party is corrupted, or if the adversary successfully guessed the player's password (the session is then marked as *compromised*), the adversary is granted the right to fully determine its session key. There is in fact nothing lost by allowing it to determine the key. In case of a wrong guess (the session is then marked as *interrupted*), the two players are given independently-chosen random keys. A session that is nor *compromised* nor *interrupted* is called *fresh*, which is its initial status.

Finally notice that the functionality is not in charge of providing the passwords to the participants. The passwords are chosen by the environment which then hands them to the parties as inputs. This

[1] The most efficient PAKE scheme so far (using completely different techniques) is the recent Asiacrypt paper [JR15], which relies on QA-NIZK.

The functionality $\mathcal{F}_{\text{pwKE}}$ is parameterized by a security parameter $\mathfrak{K}$. It interacts with an adversary $\mathcal{S}$ and a set of parties $P_1,\ldots,P_n$ via the following queries:

- **Upon receiving a query $(\textit{NewSession}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{pw})$ from party P_i:**
  Send $(\textit{NewSession}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to $\mathcal{S}$. If this is the first *NewSession* query, or if this is the second *NewSession* query and there is a record $(\mathsf{sid}, \mathsf{ssid}, P_j, P_i, \mathsf{pw}')$, then record $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{pw})$ and mark this record *fresh*.
- **Upon receiving a query $(\textit{TestPwd}, \mathsf{sid}, \mathsf{ssid}, P_i, \mathsf{pw}')$ from the adversary S:**
  If there is a record of the form (P_i, P_j, pw) which is *fresh*, then do: If $\mathsf{pw} = \mathsf{pw}'$, mark the record *compromised* and reply to $\mathcal{S}$ with "correct guess". If $\mathsf{pw} \neq \mathsf{pw}'$, mark the record *interrupted* and reply with "wrong guess".
- **Upon receiving a query $(\textit{NewKey}, \mathsf{sid}, \mathsf{ssid}, P_i, \mathsf{sk})$ from the adversary S:**
  If there is a record of the form $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{pw})$, and this is the first *NewKey* query for P_i, then:
  - If this record is *compromised*, or either P_i or P_j is corrupted, then output $(\mathsf{sid}, \mathsf{ssid}, \mathsf{sk})$ to player P_i.
  - If this record is *fresh*, and there is a record (P_j, P_i, pw') with $\mathsf{pw}' = \mathsf{pw}$, and a key sk' was sent to P_j, and (P_j, P_i, pw) was *fresh* at the time, then output $(\mathsf{sid}, \mathsf{ssid}, \mathsf{sk}')$ to P_i.
  - In any other case, pick a new random key sk' of length $\mathfrak{K}$ and send $(\mathsf{sid}, \mathsf{ssid}, sk')$ to P_i.

  Either way, mark the record $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{pw})$ as *completed*.

Figure 6.3: Ideal Functionality for PAKE $\mathcal{F}_{\text{pwKE}}$

guarantees security even in the case where two honest players execute the protocol with two different passwords: This models, for instance, the case where a user mistypes its password. It also implies that the security is preserved for all password distributions (not necessarily the uniform one) and in all situations where the password, are related passwords, are used in different protocols. Also note that allowing the environment to choose the passwords guarantees forward secrecy.

In case of corruption, the adversary learns the password of the corrupted player, after the *NewKey*-query, it additionally learns the session key.

6.2.2 High Level Construction

Our goal is to build symmetric one-round PAKE. In other words, we want to make PAKE protocols, where both user behave in the same way, and just need to them one flow without respecting a particular order.

Throughout different papers [BBC+13b, ABB+13, BC16], we proposed protocols following a basic idea. Assuming a common reference string containing the public key of a commitment, each user will then commit to his password pw in $\mathbf{C}$, keep the associated witness w, and provide the projection key hp for a KV-SPHF for the language of a valid commitment of said password. This is detailed in Figure 6.4, page 37.

CRS: $\mathsf{crs} \xleftarrow{\$} \mathsf{SetupCom}(1^{\mathfrak{K}})$.
Protocol execution by P_i with pw_i:

1. P_i generates $\mathsf{hk}_i \xleftarrow{\$} \mathsf{HashKG}(\mathscr{L}_{\mathsf{pw}_i})$, $\mathsf{hp}_i \leftarrow \mathsf{ProjKG}(\mathsf{hk}_i, \mathscr{L}_{\mathsf{pw}_i})$
2. P_i computes $(\mathbf{C}_i, w_i) = \mathsf{Com}^{\ell_i}(\mathsf{crs}, \mathsf{pw}_i, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$ with $\ell_i = (\mathsf{sid}, P_i, P_j, \mathsf{hp}_i)$
3. P_i sends $\mathsf{hp}_i, \mathbf{C}_i$ to P_j, stores hk_i, w_i, completely erases everything else.

Key computation: Upon receiving $\mathsf{hp}_j, \mathbf{C}_j$ from P_j

1. P_i computes $H'_i \leftarrow \mathsf{ProjHash}(\mathsf{hp}_j, (\mathscr{L}_{\mathsf{pw}_i}, \ell_i, \mathbf{C}_i))$
   and $H_j \leftarrow \mathsf{Hash}(\mathsf{hk}_i, (\mathscr{L}_{\mathsf{pw}_i}, \ell_j, \mathbf{C}_j))$ with $\ell_j = (\mathsf{sid}, P_j, P_i, \mathsf{hp}_j)$
2. P_i computes $\mathsf{sk}_i = H'_i \oplus H_j$ and erases everything else, except pw_i.

Figure 6.4: One-Round PAKE from a Commitment with its KV-SPHF

We proposed to instantiate such framework with various primitives to achieve different level of security

/ efficiency.

We started [BBC+13b] by using a simple Cramer Shoup Encryption as commitment. This leads to a very efficient PAKE (each user sending 6 group elements), secure in the BPR model [BPR00] which is not totally UC because it cannot handle corruption.

We then proceeded [ABB+13], to supersede Cramer Shoup with a commitment *à la* Haralambiev, we showed that this commitment was both extractable, equivocable, and compatible with SPHF. This lead to a One-Round UC-PAKE with adaptive corruptions, however we needed to do a bit per bit commitment of the password, it then provided both an upper bound of the password length to the adversary, and was still inefficient as linear in the password length.

Finally, we managed [BC16] through the introduction of Structure-Preserving SPHF, to supersede this commitment with the UC-Commitment from [FLM11] to obtain a constant-size UC-PAKE. Sadly, simultaneously [JR15] proposed another approach and obtained a slightly more efficient scheme.

Table 6.1: Comparison with existing UC-secure PAKE schemes where $|\text{password}| = m$

	Adaptive	One-round	Communication complexity	Assumption
[ACP09]	✓	✗	$2 \times (2m + 22m\mathfrak{K}) \times \mathbb{G} + \text{OTS}$	DDH
[KV11]	✗	✓	$\approx 2 \times 70 \times \mathbb{G}$	DLIN
[BBC+13b]	✗	✓	$2 \times 6 \times \mathbb{G}_1 + 2 \times 5 \times \mathbb{G}_2$	SXDH
[ABB+13]	✓	✓	$2 \times 10m \times \mathbb{G}_1 + 2 \times m \times \mathbb{G}_2$	SXDH
[JR15]	✓	✓	$6 \times \mathbb{G}_1 + 2 \times \mathbb{G}_2$	SXDH
[BC16]	✓	✓	$2 \times 4 \times \mathbb{G}_1 + 2 \times 5 \times \mathbb{G}_2$	SXDH

6.3 Verifier-based PAKE

The problem with PAKE protocols is that in case of corruptions, the adversary has a direct access to the password in plaintext. Often a PAKE is going to be between a user, and a server, in this context the server is more prone to attacks as it will normally store several passwords. To avoid problems in case of *server compromise*, the concept of verifier-based PAKE was introduced, the client still owns the password pw, however the server now only knows a one-way function of the password $f(\mathsf{pw})$ (for example g^{pw} or $\mathcal{H}(\mathsf{pw})$). This way in case of a leak, an adversary does not have directly access to the passwords.

The idea is to process in two (simultaneous) steps:

- On one hand, they do a PAKE for the password $f(\mathsf{pw})$. This is enough to tell the user that the server in front of him knows the required function of the password, however in case of a leak of the database, an adversary would be able to impersonate the user when authenticating
- On the other hand, the user also needs to prove he knows the password (pre-image of $f(\mathsf{pw})$). In case of a password stored as g^{pw}, the server can pick a discrete logarithm defining a generator $h = g^{\alpha}$, and then asking the user to send h^{pw} and (via an SPHF for example) show that $h, g, h^{\mathsf{pw}}, g^{\mathsf{pw}}$ is a DDH tuple.

6.4 DPAKE

Another technique to protect against server compromise is simply to share the passwords between several servers. This alleviate the risk that in case of a database breach, adversary then further run an offline attack to find a preimage.

We considered in [BCV16], an alternative approach inspired by the multi-party computation paradigm (and first suggested by Ford and Kaliski [FK00]). The password database on the server side is somehow shared among two servers (or more, but we focus here on two for sake of simplicity), and authentication requires a distributed computation involving the client – who still does not need an additional cryptographic device capable of storing high-entropy secret keys – and the two servers who will use some additional shared secret information. The interaction is performed using a *gateway* that does not know any secret information and ends up in the gateway and the client sharing a common key. The lifetime of the protocol is divided into distinct periods (for simplicity, one may think of these time periods as being of equal length; e.g. one day) and at the beginning of each period, the two servers interact and update their sharing of the password database. Similarly to proactive schemes in multi-party computation, we allow the adversary multiple corruptions of each server, limiting only the corruptions to one server for each period. The user does not need to update his password nor to perform any kind of computations

and its interaction with the two servers (performed using the gateway) remains the same for the lifetime of the protocol. In this scenario, even if a server compromise is doable, the secret exposure is not valuable to the adversary since it reveals only a share of the password database and does not permit to run an offline dictionary attack.

Ford and Kaliski [FK00] were the first to propose to distribute the capability to test passwords over multiple servers. Building on this approach, several such protocols were subsequently proposed in various settings (*e.g.* [Jab01, MSJ02, BJKS03, DG03, DG06, SK05, KMTG05, KMTG12, ACFP05, KM14]), some of these solutions (like [BJKS03]) are even commercially available. Recently, Camenisch, Enderlein and Neven [CEN15] revisited this approach and proposed a scheme in the universal composability framework [Can01] (which has obvious advantages for password-based protocols since users often use related passwords for many providers). Camenisch *et al.* gave interesting details about the steps that need to be taken when a compromise actually occurs. Unfortunately, due to the inherent difficulties of construction of the simulator in the universal composability framework, their scheme requires users and servers have to perform a few hundred exponentiations each, which is far from being practical¡

6.4.1 Constructions

Our first construction uses a similar approach to the aforementioned schemes from [Jab01, MSJ02, BJKS03, DG06, SK05, KMTG12, ACFP05, KM14]: the user generates information theoretic shares of his password and sends them to the servers. In the authentication phase, the parties run a dedicated protocol to verify that the provided password equals the priorly shared one. Our solution then consists in some sort of three-party PAKE, in which (1) the user implicitly checks (using a smooth projective hash function) that its password is indeed the sum of the shares owned by the two servers, and (2) each server implicitly checks that its share is the difference of the password owned by the user and the share owned by the other server. Contrary to the popular approach initiated in [KOY01, GL03] for PAKE, we cannot use two smooth projective hash functions (one for the client and one for the server) so we propose a technique in order to combine in a secure way six smooth projective hash functions. This new method (which may be of independent interest) allows us to prove the security of this construction under classical cryptographic assumptions (namely the DDH assumption) in the standard security model from [KMTG12] (without any idealized assumptions).

The main weakness of this first solution is that at each time period, the servers have to refresh the information-theoretic sharing of the password of all users. This can be handled easily using well-known techniques from proactive multi-party computation but if the number of users is large, this can be really time-consuming (in particular if the time period is very short). Our second construction (which is the main contribution of the paper) is built on the ideas from the first one but passwords are now encrypted using a public-key encryption scheme where the corresponding secret key is shared among the servers. At the beginning of each time period, the servers only need to refresh the sharing of this secret key but the password database is not modified (and can actually be public). Password verification and the authenticated key exchange is then carried out without ever decrypting the database. A secure protocol is run to verify that the password sent by the user matches the encrypted password. It is similar to the protocol we design for the first construction except that the user encrypts its password and the parties implicitly check (using in this case five smooth projective hash functions) that the message encrypted in this ciphertext is the same as the message encrypted in the database (using the secret key shared upon the servers). Both constructions consist in only two flows (one from the client and one from the servers) and a (private) flow from the servers to the gateway.

Distributed PAKE.

In a distributed PAKE system, we consider as usual a client (owning a password) willing to interact with a gateway, such as a website. The difference compared to a non-distributed system is that the gateway itself and interacts with two servers, and none of the three owns enough information to be able to recover the passwords of the clients on its own[2]. Such a scheme is correct if the interaction between a client with a correct password and the gateway succeeds. An honest execution of a distributed PAKE protocol should result in the client holding a session key K_{U} and the gateway holding a session key $K_{\mathsf{G}} = K_{\mathsf{U}}$.

We propose in this paper two settings that describe well this situation. In a first setting, we consider that the passwords of the clients are shared information-theoretically between the servers, such as $\pi =$

[2]Note that the gateway can be merged with one server, or for a web interface it will often be one of several mirrors with no secret.

$\pi_1 + \pi_2$ (if the password π belongs to an appropriate group) or with the help of any secret sharing protocol. At the beginning of each time period, the shares are updated, in a probabilistic way, using a public function *Refresh*, depending on the sharing protocol used.

In a second setting, we consider that the gateway owns a database of encrypted passwords (which can be considered public), and the servers each own a share of the corresponding private keys (obtained by a secret sharing protocol). Again, at the beginning of each time period, the shares are updated, in a probabilistic way, using a public function *Refresh*, depending on the sharing protocol used.

If we stay outside the universal composability framework, the *Refresh* procedure can be handled easily using classical techniques from computational proactive secret sharing (see [OY91, HJKY95] for instance).

In this case, we consider the classical model [BPR00] for authenticated key-exchange, adapted to the two-server setting by [ACFP05, KMTG12]. In the latter model, the authors assume that every client in the system shares its password with exactly two servers. We loosen this requirement here, depending on the setting considered.

6.4.2 Simple Protocol

In this first setting, we consider a client U owning a password π and willing to interact with a gateway G. The gateway interacts with two servers S_1 (owning π_1) and S_2 (owning π_2), such that $\pi = \pi_1 + \pi_2$. It should be noted that only the client's password is assumed to be small and human-memorable. The two "passwords" owned by the servers can be arbitrarily big. The aim of the protocol is to establish a shared session key between the client and the gateway.

A simple solution to this problem consists in considering some sort of three-party PAKE, in which the client implicitly checks (using an SPHF) whether its password is the sum of the shares owned by the two servers, and the servers implicitly check (also using an SPHF) whether their share is the difference of the password owned by the client and the share owned by the other server. For sake of simplicity, we denote the client U as S_0 and its password π as π_0.

Main Idea of the Construction. In our setting, we denote by pw_b a group element, this can either be g^{π_b} or more cleanly $G(\pi_b)$ where G is a reversible group embedding function. The main idea of the protocol is depicted on Figure 6.5, page 40. For sake of readability, the participants which have a real role in the computations are directly linked by arrows in the picture, but one should keep in mind that all the participants (U, S_1 and S_2) only communicate with G, which then broadcasts all the messages.

In a classical SPHF-based two-party key-exchange between U and G, the client and the gateway simply checks whether they sent valid Cramer Shoup encryption of the same password.

Here, since U owns $\mathsf{pw}_0 = \mathsf{pw}_1 \cdot \mathsf{pw}_2$, it gets a little trickier. Also we assume that U does not know the precise decomposition $\mathsf{pw}_1, \mathsf{pw}_2$, which means that we can not simply run two two-party PAKE one between U and $\mathcal{S}_1$ and one between U and $\mathcal{S}_2$. In fact, everything is symmetrical in the scheme, and one can interchange U with any server without altering their role in the scheme. So, taking that into account, we need to run 6 SPHF in total to globally ensure the correctness and smoothness of the scheme. Where, considering the SPHFs for the pair $(\mathcal{S}_i, \mathcal{S}_j)$ implies that if and only if everything was computed honestly, then one gets the equalities $H_{i,j}(\mathsf{pw}_i/\mathsf{pw}_j)^{\lambda_i} = H'_{i,j}$ and $H_{j,i}(\mathsf{pw}_j/\mathsf{pw}_i)^{\lambda_j} = H'_{j,i}$.

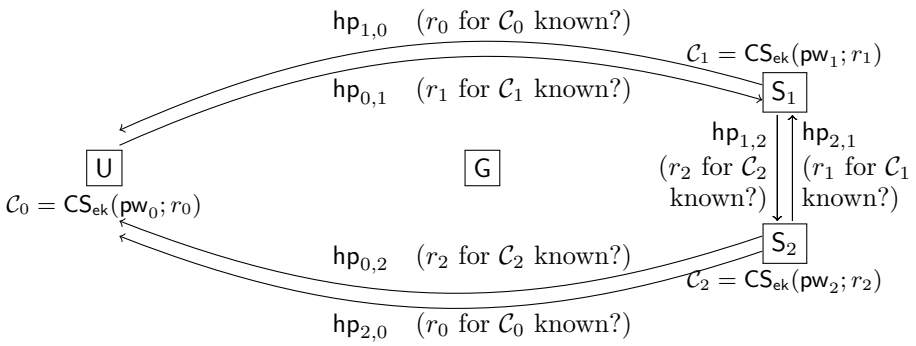


Figure 6.5: Main idea of the Simple approach

6.4.3 Login procedure

- Each participant S_b picks r_b at random and computes a Cramer-Shoup encryption of its password $\mathcal{C}_b = \mathsf{CS}_{\mathsf{ek}}(\mathsf{pw}_b; r_b)$, with $v_b = cd^{\xi_b}$.
  It also chooses, for $i \in \{0,1,2\} \setminus \{b\}$, a random hash key $\mathsf{hk}_{b,i} = (\eta_{b,i}, \gamma_{b,i}, \theta_{b,i}, \lambda_b, \kappa_{b,i})$ and sets $\mathsf{hp}_{b,i} = (\mathsf{hp}_{b,i;1}, \mathsf{hp}_{b,i;2}) = ({g_1}^{\eta_{b,i}} {g_2}^{\theta_{b,i}} h^{\lambda_b} c^{\kappa_{b,i}}, {g_1}^{\gamma_{b,i}} d^{\kappa_{b,i}})$ as the projection key intended to the participant S_i.
  It sends $(\mathcal{C}_b, (\mathsf{hp}_{b,i})_{i \in \{0,1,2\} \setminus \{b\}})$ to the gateway G, which broadcasts these values to the other participants.
- After receiving the first flow from the servers, the client computes $H'_{i,0} = {\mathsf{hp}_{i,0;1}}^{r_0} {\mathsf{hp}_{i,0:2}}^{\xi_0 r_0}$ for $i \in \{1,2\}$. It also computes $H_0 = H_{0,1} \cdot H_{0,2} \cdot {\mathsf{pw}_0}^{\lambda_0}$, and sets its session key K_{U} as $K_{\mathsf{U}} = K_0 = H'_{1,0} \cdot H'_{2,0} \cdot H_0$.
  After receiving the first flow from the other server and the client, the server S_b computes, for $i \in \{0, 3-b\}$, $H'_{i,b} = {\mathsf{hp}_{i,b;1}}^{r_b} {\mathsf{hp}_{i,b;2}}^{\xi_b r_b}$. It also computes $H_b = H_{b,0}/(H_{b,3-b} \cdot {\mathsf{pw}_b}^{\lambda_b})$, and sets its partial key K_b as $K_b = H'_{0,b} \cdot H'_{3-b,b} \cdot H_b$. It privately sends this value K_b to the gateway G.
- The gateway finally sets $K_{\mathsf{G}} = K_1 \cdot K_2$.

6.4.4 Efficient Version

Now, instead of assuming that the servers possess share of the passwords, we assume they possess a share of a decryption key, and that the gateway possesses the whole database of encrypted password, the client U still owns a password π. The aim of the protocol is to establish a shared session key between the client and the gateway.

The idea is similar to the protocol described before, except that only the client needs to compute a ciphertext, the other ciphertext being publicly available from the database. The participants implicitly check (using several SPHF) that the message encrypted in the ciphertext of the client is the same as the message encrypted in the database (using the secret key shared upon the servers).

Main Idea of the Construction. Again, we denote the client U as S_0 and its password π as π_0. For simplicity we assume the database contains ElGamal encryptions of each ciphertext $\mathsf{pw}_{\mathsf{U}_i}$, under randomness s_{U_i}: $\mathcal{C}^{db}_{\mathsf{U}_i} = \mathsf{EG}_{\mathsf{pk}}(\mathsf{pw}_{\mathsf{U}_i}; s_{\mathsf{U}_i}) = (h^{s_{\mathsf{U}_i}} \mathsf{pw}_{\mathsf{U}_i}, g^{s_{\mathsf{U}_i}})$, so that here, $\mathcal{C}^{db}_{\mathsf{U}} = \mathsf{EG}_{\mathsf{pk}}(\mathsf{pw}_{\mathsf{U}}; s_{\mathsf{U}}) = (h^{s_{\mathsf{U}}} \mathsf{pw}_{\mathsf{U}}, g^{s_{\mathsf{U}}})$. The client computes a Cramer-Shoup encryption of its password: $\mathcal{C}_0 = \mathsf{CS}_{\mathsf{ek}}(\mathsf{pw}_0; r_0) = (u_1, u_2, e, v)$ with $v = cd^{\xi}$. The execution of the protocol should succeed if these encryptions are correct and $\mathsf{pw}_0 = \mathsf{pw}_{\mathsf{U}}$. Recall that the server $\mathcal{S}_i$ knows α_i such that $\alpha = \alpha_1 + \alpha_2$ is the decryption key of the ElGamal encryption.

The main idea is depicted on Figure 6.6, page 42. Again, for readability, the participants which have a real role in the computations are directly linked by arrows in the picture instead of communicating via the G.

In a classical SPHF-based two-party key-exchange between U and G, the gateway would check if $\mathcal{C}_0$ is a valid Cramer-Shoup encryption of pw_{U}. Since here the password pw_{U} is unknown to the servers S_1 and S_2, this is done in our setting by two SPHF, using $\mathsf{hp}^{\mathrm{CS}}_1$ (sent by S_1) and $\mathsf{hp}^{\mathrm{CS}}_2$ (sent by S_2), where the servers use the first term of the public encryption $\mathcal{C}^{\mathrm{DB}}_{\mathsf{U}}$ ($h^{s_{\mathsf{U}}} \mathsf{pw}_{\mathsf{U}}$) in order to cancel the unknown pw_{U}.

In a classical SPHF-based two-party key-exchange between U and G, the client would also check whether $\mathcal{C}^{\mathrm{DB}}_{\mathsf{U}}$ is a valid El Gamal encryption of its password pw_0, i.e. whether the gateway knows a witness for its ciphertext $\mathcal{C}^{\mathrm{DB}}_{\mathsf{U}}$ (s_{U} in the usual constructions, α here). Since α is unknown to the gateway, this is done in our setting by the combination of three SPHF, using $\mathsf{hp}^{\mathrm{EG}}_0$ (sent by the client), $\mathsf{hp}^{\mathrm{EG}}_1$ (sent by S_1) and $\mathsf{hp}^{\mathrm{EG}}_2$ (sent by S_2). These three SPHF allow the client and the servers to implicitly check that the servers know α_1 and α_2 such that $\mathcal{C}^{\mathrm{DB}}_{\mathsf{U}}$ can be decrypted (using the decryption key $\alpha = \alpha_1 + \alpha_2$) to the same password pw_0 than the one encrypted in $\mathcal{C}_0$ sent by the client. Formally, the languages checked are as follows:

- by the client:
  $\mathcal{C}^{\mathrm{DB}}_{\mathsf{U}} \in \mathscr{L}_0 = \{\mathcal{C} = (e,u) \in \mathbb{G}^2 \mid \exists \alpha \in \mathbb{Z}_p \text{ such that } h = g^{\alpha} \text{ and } e/u^{\alpha} = \mathsf{pw}_0\}$
- by server $\mathcal{S}_i$ (with respect to the client $\mathcal{S}_0$ and server $\mathcal{S}_j$):
  $\mathcal{C}_0 \in \mathscr{L}_{i,0} = \{\mathcal{C} = (u_1, u_2, e, v) \in \mathbb{G}^4 \mid \exists r \in \mathbb{Z}_p \text{ such that } \mathcal{C} = \mathsf{CS}_{\mathsf{ek}}(\mathsf{pw}_{\mathsf{U}}; r) \text{ and } \mathcal{C}^{\mathrm{DB}}_{\mathsf{U}} \in \mathscr{L}_{i,j} = \{\mathcal{C} = (e,u) \in \mathbb{G}^2 \mid \exists \alpha_j \in \mathbb{Z}_p \text{ such that } h = g^{\alpha_i + \alpha_j}\} \text{ and } e/u^{\alpha_i + \alpha_j} = \mathsf{pw}_{\mathsf{U}}\}$

but they cannot be checked directly by a unique SPHF since the value pw_{U} appearing in the languages is unknown to the verifier $\mathcal{S}_i$. Rather, the server $\mathcal{S}_i$ will use the first term of the public encryption $\mathcal{C}^{\mathrm{DB}}_{\mathsf{U}}$ ($h^{s_{\mathsf{U}}} \mathsf{pw}_{\mathsf{U}}$) in order to cancel this unknown pw_{U}. To achieve this goal, we combine the five SPHF described

to globally ensure the correctness (each one remaining smooth and pseudo-randomness), as described in the next part.

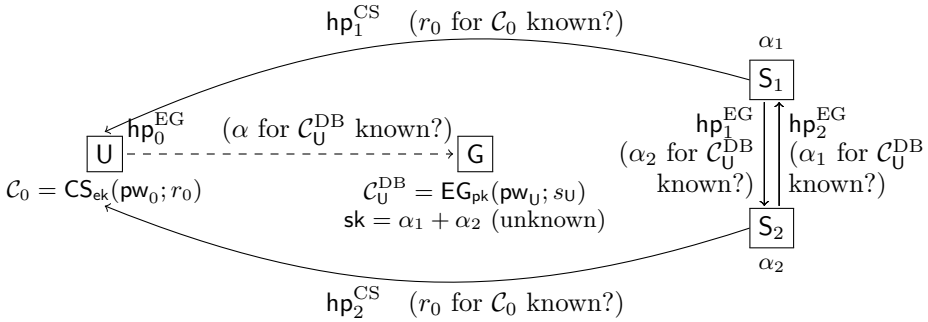


Figure 6.6: Main idea of the construction

6.5 Secret Handshake

The concept of *Secret Handshakes* has been introduced in 2003 by Balfanz, Durfee, Shankar, Smetters, Staddon and Wong [BDS+03] (see also [JL09b, AKB07]). It allows two members of the same group to identify each other secretly, in the sense that each party reveals his affiliation to the other only if they are members of the same group. At the end of the protocol, the parties can set up an ephemeral session key for securing further communication between them and an outsider is unable to determine if the handshake succeeded.

When compared to a PAKE, we do not require equality of the signatures possessed by the users but an equality of the verification key vk under which those signatures are valid (this means that they were both *accredited* by the same organization).

To achieve such scheme, we can proceed as with the PAKE, except with a different SPHF as shown in Figure 6.7, page 42. For simplicity we assume the associated SPHF (for the language of committed value fulfilling the verification equation of the signature under the parameter vk) is KV. This may not always be the case, if the verification equation has a pairing between two parts of the signature, and those two parts have to be hidden [3].

CRS: $\mathsf{crs} \xleftarrow{\$} \mathsf{SetupCom}(1^{\mathfrak{K}})$.
Protocol execution by P_i with σ_i valid under vk_i:

1. P_i generates $\mathsf{hk}_i \xleftarrow{\$} \mathsf{HashKG}(\mathscr{L}_{\mathsf{vk}_i})$, $\mathsf{hp}_i \leftarrow \mathsf{ProjKG}(\mathsf{hk}_i, \mathscr{L}_{\mathsf{vk}_i})$
2. P_i computes $(\mathbf{C}_i, w_i) = \mathsf{Com}^{\ell_i}(\mathsf{crs}, \sigma_i, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$ with $\ell_i = (\mathsf{sid}, P_i, P_j, \mathsf{hp}_i)$
3. P_i sends $\mathsf{hp}_i, \mathbf{C}_i$ to P_j, stores hk_i, w_i, completely erases everything else.

Key computation: Upon receiving $\mathsf{hp}_j, \mathbf{C}_j$ from P_j

1. P_i computes $H'_i \leftarrow \mathsf{ProjHash}(\mathsf{hp}_j, (\mathscr{L}_{\mathsf{vk}_i}, \ell_i, \mathbf{C}_i))$
   and $H_j \leftarrow \mathsf{Hash}(\mathsf{hk}_i, (\mathscr{L}_{\mathsf{vk}_i}, \ell_j, \mathbf{C}_j))$ with $\ell_j = (\mathsf{sid}, P_j, P_i, \mathsf{hp}_j)$
2. P_i computes $\mathsf{sk}_i = H'_i \oplus H_j$ and erases everything else, except σ_i.

Figure 6.7: One-Round Secret Handshake from a Commitment with its KV-SPHF

⋆

[3] This problem mostly arises in compact Structure Preserving Signatures, as they can not be efficiently randomized.

Chapter 7

Asymmetric Constructions (OLBE)

Contents

Continuing our cryptographic zoo; we are now considering a series of asymmetric primitives. From a very high level, we consider protocols between a server and a user, where the user wants to retrieve obliviously a message from a server. Of course, the server might want to allow it under maybe some special condition.

Such protocols naturally arose from the notion of Oblivious Transfer from [Rab81]. We proposed in [BCG16] a generalization superseding all the previous definition, together with a generic way to achieve it.

7.1 OLBE

Similarly to LAKE, we wanted to generalize the family of protocols around Oblivious Transfer, Oblivious-Signature Based Envelope, and so one, thus in [BCG16], we introduced the concept of Oblivious Language-Based Envelope. In those protocols,a sender $\mathcal{S}$ wants to send one or several private messages (up to $n_{\max} \leq n$) among $(m_1, \ldots, m_n) \in (\{0,1\}^\ell)^n$ to a recipient $\mathcal{R}$ in possession of a tuple of words $\chi = (\chi_{i_1}, \ldots, \chi_{i_{n_{\max}}})$ such that some of the words χ_{i_j} may belong to the corresponding language $\mathscr{L}_{i_j}$. More precisely, the receiver gets each m_{i_j} as soon as $\chi_{i_j} \in \mathscr{L}_{i_j}$ with the requirement that he gets at most $n_{\max}$ messages. In such a scheme, the languages $(\mathscr{L}_1, \ldots, \mathscr{L}_n)$ are assumed to be a trapdoor collection of languages, publicly verifiable and self-randomizable.

As we consider simulation-based security (in the UC framework), we allow a simulated setup SetupT to be run instead of the classical setup Setup in order to allow the simulator to possess some trapdoors. Those two setup algorithms should be indistinguishable.

Oblivious Language-Based Envelope
⌜An OLBE scheme is defined by four algorithms ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Samp}, \mathsf{Verif}$), and one interactive protocol $\mathsf{Protocol}\langle\mathcal{S}, \mathcal{R}\rangle$:

- $\mathsf{Setup}(1^{\mathfrak{K}})$, where $\mathfrak{K}$ is the security parameter, generates the global parameters param, among which the numbers n and $n_{\max}$;

or $\mathsf{SetupT}(1^{\mathfrak{K}})$, where $\mathfrak{K}$ is the security parameter, additionally allows the existence[1] of a trapdoor tk for the collection of languages $(\mathscr{L}_1, \ldots, \mathscr{L}_n)$.

- $\mathsf{KeyGen}(\mathsf{param}, \mathfrak{K})$ generates, for all $i \in \{1, \ldots, n\}$, the description of the language $\mathscr{L}_i$ (as well as the language key $\mathsf{sk}_{\mathscr{L}_i}$ if need be). If the parameters param were defined by SetupT, this implicitly also defines the common trapdoor tk for the collection of languages $(\mathscr{L}_1, \ldots, \mathscr{L}_n)$.
- $\mathsf{Samp}(\mathsf{param}, I)$ or $\mathsf{Samp}(\mathsf{param}, I, (\mathsf{sk}_{\mathscr{L}_i})_{i\in I})$ such that $I \subset \{1, \ldots, n\}$ and $|I| = n_{\max}$, generates a list of words $(\chi_i)_{i\in I}$ such that $\chi_i \in \mathscr{L}_i$ for all $i \in I$;
- $\mathsf{Verif}_i(\chi_i, \mathscr{L}_i)$ checks whether χ_i is a valid word in the language $\mathscr{L}_i$. It outputs 1 if the word is valid, 0 otherwise;
- $\mathsf{Protocol}\langle \mathcal{S}((\mathscr{L}_1, \ldots, \mathscr{L}_n), (m_1, \ldots, m_n)), \mathcal{R}((\mathscr{L}_1, \ldots, \mathscr{L}_n), (\chi_i)_{i\in I})\rangle$, which is executed between the sender $\mathcal{S}$ with the private messages $(m_1, \ldots, P_n)$ and corresponding languages $(\mathscr{L}_1, \ldots, \mathscr{L}_n)$, and the recipient $\mathcal{R}$ with the same languages and the words $(\chi_i)_{i\in I}$ with $I \subset \{1, \ldots, n\}$ and $|I| = n_{\max}$, proceeds as follows. For all $i \in I$, if the algorithm $\mathsf{Verif}_i(\chi_i, \mathscr{L}_i)$ returns 1, then $\mathcal{R}$ receives m_i, otherwise it does not. In any case, $\mathcal{S}$ does not learn anything.

⌟

7.1.1 Security Properties and Ideal Functionality of OLBE

Since we aim at proving the security in the universal composability framework, we now describe the corresponding ideal functionality (depicted in Figure 7.1). However, in order to ease the reading, we first list the security properties required:

- *correct*: the protocol actually allows $\mathcal{R}$ to learn $(m_i)_{i\in I}$, whenever $(\chi_i)_{i\in I}$ are valid words of the languages $(\mathscr{L}_i)_{i\in I}$, where $I \subset \{1, \ldots, n\}$ and $|I| = n_{\max}$;
- *semantically secure* (*sem*): the recipient learns nothing about the input m_i of $\mathcal{S}$ if it does not use a word in $\mathscr{L}_i$. More precisely, if $\mathcal{S}_0$ owns $m_{i,0}$ and $\mathcal{S}_1$ owns $m_{i,1}$, the recipient that does not use a word in $\mathscr{L}_i$ cannot distinguish between an interaction with $\mathcal{S}_0$ and an interaction with $\mathcal{S}_1$ even if the receiver has seen several interactions
$$\langle \mathcal{S}((\mathscr{L}_1, \ldots, \mathscr{L}_n), (m_1, \ldots, m_n)), \mathcal{R}((\mathscr{L}_1, \ldots, \mathscr{L}_n), (\chi'_j)_{j\in I})\rangle$$
with valid words $\chi'_i \in \mathscr{L}_i$, and the same sender's input m_i;
- *escrow free* (*oblivious with respect to the authority*): the authority corresponding to the language $\mathscr{L}_i$ (owner of the language secret key $\mathsf{sk}_{\mathscr{L}_i}$ – if it exists), playing as the sender or just eavesdropping, is unable to distinguish whether $\mathcal{R}$ used a word χ_i in the language $\mathscr{L}_i$ or not. This requirement also holds for anyone holding the trapdoor key tk.
- *semantically secure w.r.t. the authority* (*sem**): after the interaction, the trusted authority (owner of the language secret keys if they exist) learns nothing about the values $(m_i)_{i\in I}$ from the transcript of the execution. This requirement also holds for anyone holding the trapdoor key tk.

Moreover, the Setups should be indistinguishable and it should be infeasible to find a word belonging to two or more languages without the knowledge of tk.

The functionality $\mathcal{F}_{\mathrm{OLBE}}$ is parametrized by a security parameter $\mathfrak{K}$ and a set of languages $(\mathscr{L}_1, \ldots, \mathscr{L}_n)$ along with the corresponding public verification algorithms $(\mathsf{Verif}_1, \ldots, \mathsf{Verif}_n)$. It interacts with an adversary $\mathcal{S}$ and a set of parties $\mathfrak{P}_1, \ldots, \mathfrak{P}_N$ via the following queries:

- **Upon receiving from party $\mathfrak{P}_i$ an input of the form $(\boldsymbol{Send}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1, \ldots, m_n))$** , with $m_k \in \{0,1\}^{\mathfrak{K}}$ for all k: record the tuple $(\mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1, \ldots, m_n))$ and reveal $(Send, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j)$ to the adversary $\mathcal{S}$. Ignore further *Send*-message with the same ssid from $\mathfrak{P}_i$.
- **Upon receiving an input of the form $(\boldsymbol{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (\chi_i)_{i\in I})$ where $I \subset \{1, \ldots, n\}$ and $|I| = n_{\max}$ from party $\mathfrak{P}_j$**: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1, \ldots, m_n))$ is not recorded. Otherwise, reveal $(Receive, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j)$ to the adversary $\mathcal{S}$ and send the message $(Received, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m'_k)_{k\in I})$ to $\mathfrak{P}_j$ where $m'_k = m_k$ if $\mathsf{Verif}_k(\chi_k, \mathscr{L}_k)$ returns 1, and $m'_k = \perp$ otherwise. Ignore further *Received*-message with the same ssid from $\mathfrak{P}_j$.

Figure 7.1: Ideal Functionality for Oblivious Language-Based Envelope $\mathcal{F}_{\mathrm{OLBE}}$

The ideal functionality is parametrized by a set of languages $(\mathscr{L}_1, \ldots, \mathscr{L}_n)$. We use the simple UC framework for simplicity (this enables us to get rid of *Sent* and *Received* queries from the adversary

[1] The specific trapdoor will depend on the languages and be computed in the KeyGen algorithm.

since the delayed outputs are automatically considered in this simpler framework: We implicitly let the adversary determine if it wants to acknowledge the fact that a message was indeed sent). The first step for the sender (*Send* query) consists in telling the functionality he is willing to take part in the protocol, giving as input his intended receiver and the messages he is willing to send (up to $n_{\max}$ messages). For the receiver, the first step (*Receive* query) consists in giving the functionality the name of the player he intends to receive the messages from, as well as his words. If the word does belong to the language, the receiver recovers the sent message, otherwise, he only gets a special symbol $\perp$.

7.1.2 Generic UC-Secure Instantiation of OLBE with Adaptive Security

For the sake of clarity, we focuses on the specific case where $n_{\max} = 1$. This is the most classical case in practice. In order to get a generic protocol in which $n_{\max} > 1$, one simply has to run $n_{\max}$ protocols in parallel.

This modifies the algorithms Samp and Verify as follows: $\mathsf{Samp}(\mathsf{param}, \{i\})$ or $\mathsf{Samp}(\mathsf{param}, \{i\}, \{\mathsf{sk}_{\mathscr{L}_i}\})$ generates a word $\mathcal{x} = \mathcal{x}_i \in \mathscr{L}_i$ and $\mathsf{Verif}_j(\mathcal{x}, \mathscr{L}_j)$ checks whether $\mathcal{x}$ is a valid word in $\mathscr{L}_j$.

Let us introduce our protocol OLBE: we will call $\mathcal{R}$ the receiver and $\mathcal{S}$ the sender. If $\mathcal{R}$ is an honest receiver, then he knows a word $\mathcal{x} = \mathcal{x}_i$ in one of the languages $\mathscr{L}_i$. If $\mathcal{S}$ is an honest sender, then he wants to send up a message among $(m_1, \ldots, m_n) \in (\{0,1\}^{\mathfrak{K}})^n$ to $\mathcal{R}$. We assume the languages $\mathscr{L}_i$ to be self-randomizable and publicly verifiable. We also assume the collection of languages $(\mathscr{L}_1, \ldots, \mathscr{L}_n)$ possess a trapdoor, that the simulator is able to find by programming the common reference string. As recalled in the previous section, this trapdoor enables him to find a word lying in the intersection of the n languages. This should be infeasible without the knowledge of the trapdoor. Intuitively, this allows the simulator to commit to all languages at once, postponing the time when it needs to choose the exact language he wants to bind to. On the opposite, if a user was granted the same possibilities, this would prevent the simulator to extract the chosen language.

We require labeled CCA-encryption scheme $\mathcal{E} = (\mathsf{Setup}_{\mathrm{cca}}, \mathsf{KeyGen}_{\mathrm{cca}}, \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}, \mathsf{Decrypt}^{\ell}_{\mathrm{cca}})$ compatible with an SPHF onto a set $\mathbb{H}$. In the KeyGen algorithm, the description of the languages $(\mathscr{L}_1, \ldots, \mathscr{L}_n)$ thus implicitly defines the languages $(\mathscr{L}_1^c, \ldots, \mathscr{L}_n^c)$ of CCA-encryptions of elements from these languages. We additionally use a key derivation function $\mathtt{KDF}$ to derive a pseudo-random bit-string $K \in \{0,1\}^{\mathfrak{K}}$ from a pseudo-random element $v \in \mathbb{H}$. One can use the Leftover-Hash Lemma [HILL99], with a random seed defined in param during the global setup, to extract the entropy from v, then followed by a pseudo-random generator to get a long enough bit-string. Many uses of the same seed in the Leftover-Hash Lemma just lead to a security loss linear in the number of extractions. We also assume the existence of a Pseudo-Random Generator (PRG) F with input size equal to the plaintext size, and output size equal to the size of the messages in the database and an $\mathsf{IND-CPA}$encryption scheme $\mathcal{E} = (\mathsf{Setup}_{\mathrm{cpa}}, \mathsf{KeyGen}_{\mathrm{cpa}}, \mathsf{Encrypt}_{\mathrm{cpa}}, \mathsf{Decrypt}_{\mathrm{cpa}})$ with plaintext size at least equal to the security parameter.

Theorem 7.1.1 *The oblivious language-based envelope scheme described in Figure 7.2 is* UC*-secure in the presence of adaptive adversaries, assuming reliable erasures, an* $\mathsf{IND-CPA}$*encryption scheme, and an* $\mathsf{IND-CCA2}$*encryption scheme admitting an SPHF on the language of valid ciphertexts of elements of* $\mathscr{L}_i$ *for all* i*, as soon as the languages are self-randomizable, publicly-verifiable and admit a common trapdoor.*

When this generic framework was proposed, it lead to the most efficient constructions of various protocols.

7.2 Oblivious Transfer

Oblivious Transfer is probably the most known primitive in this category since its introduction in 1981 by Rabin [Rab81]. A server has a database of n lines, a user wants to retrieve a line without letting the server learning which. We recall the functionality in 7.3, page 46.

Through several papers [ABB+13, BC15, BC16], we perfected our approach and proposed efficient Oblivious Transfer Scheme under various hypotheses (DQR, k-MDDH, LWE/SIS, ...). In Table 7.1, page 47, we show a progression in the efficiency of our schemes, in comparison to the state of the art. It should be noted that [BC15] manages to get rid of pairing which arguably is worth the little trade-off in efficiency. [BCG17] does not exactly follow the framework (SPHF are superseded by Quasi-Adaptive NIZK), but like [BC16] based on SP-SPHF, it manages to get rid of the extra logarithmic overhead which seemed artificial.

CRS: $\mathsf{param} \xleftarrow{\$} \mathsf{Setup}(1^{\mathfrak{K}})$, $\mathsf{param}_{\mathrm{cca}} \xleftarrow{\$} \mathsf{Setup}_{\mathrm{cca}}(1^{\mathfrak{K}})$, $\mathsf{param}_{\mathrm{cpa}} \xleftarrow{\$} \mathsf{Setup}_{\mathrm{cpa}}(1^{\mathfrak{K}})$.
Pre-flow:

1. Sender generates a key pair $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathrm{cpa}}(\mathsf{param}_{\mathrm{cpa}})$ for $\mathcal{E}$, stores sk and completely erases the random coins used by KeyGen.
2. Sender sends pk to User.

Flow From the Receiver $\mathcal{R}$:

1. User chooses a random value J, computes $R \leftarrow F(J)$ and encrypts J under pk: $c \xleftarrow{\$} \mathsf{Encrypt}_{\mathrm{cpa}}(\mathsf{pk}, J)$.
2. User computes $C \xleftarrow{\$} \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}(\mathcal{X}; r)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathcal{R}, \mathcal{S})$.
3. User completely erases J and the random coins used by $\mathsf{Encrypt}_{\mathrm{cpa}}$ and sends C and c to Sender. He also checks the validity of his words: the receiver only keeps the random coins used by $\mathsf{Encrypt}_{\mathrm{cca}}$ for the j such that $\mathsf{Verif}_j(\mathcal{X}, \mathscr{L}_j) = 1$ (since he knows they will be useless otherwise).

Flow From the Sender $\mathcal{S}$:

1. Sender decrypts $J \leftarrow \mathsf{Decrypt}_{\mathrm{cpa}}(\mathsf{sk}, c)$ and then $R \leftarrow F(J)$.
2. For all $j \in \{1, \ldots, n\}$, sender computes $\mathsf{hk}_j = \mathsf{HashKG}(\ell, \mathscr{L}_j^c, \mathsf{param})$, $\mathsf{hp}_j = \mathsf{ProjKG}(\mathsf{hk}_j, \ell, (\mathscr{L}_j^c, \mathsf{param}))$, $v_j = \mathsf{Hash}(\mathsf{hk}_j, (\mathscr{L}_j^c, \mathsf{param}), (\ell, C))$, $Q_j = m_j \oplus \mathtt{KDF}(v_j) \oplus R$.
3. Sender erases everything except $(Q_j, \mathsf{hp}_j)_{j \in \{1,\ldots,n\}}$ and sends them over a secure channel.

Message recovery:
Upon receiving $(Q_j, \mathsf{hp}_j)_{j \in \{1,\ldots,n\}}$, $\mathcal{R}$ can recover m_i by computing $m_i = Q_i \oplus \mathsf{ProjHash}(\mathsf{hp}_i, (\mathscr{L}_i^c, \mathsf{param}), (\ell, C), r) \oplus R$.

Figure 7.2: UC-Secure OLBE for One Message (Secure Against Adaptive Corruptions)

The functionality $\mathcal{F}_{(1,n)\text{-OT}}$ is parametrized by a security parameter $\mathfrak{K}$. It interacts with an adversary $\mathcal{S}$ and a set of parties $\mathfrak{P}_1, \ldots, \mathfrak{P}_N$ via the following queries:

- **Upon receiving an input $(\boldsymbol{Send}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (\boldsymbol{m_1}, \ldots, \boldsymbol{m_n}))$ from $\mathfrak{P}_i$**, with $m_k \in \{0,1\}^{\mathfrak{K}}$: record the tuple $(\mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1, \ldots, m_n))$ and reveal $(\mathit{Send}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j)$ to $\mathcal{S}$. Ignore further *Send*-message with the same ssid from $\mathfrak{P}_i$.
- **Upon receiving an input $(\boldsymbol{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, s)$ from $\mathfrak{P}_j$,** with $s \in \{1, \ldots, n\}$: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1, \ldots, m_n))$ is not recorded; otherwise reveal $(\mathit{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j)$ to $\mathcal{S}$, send $(\mathit{Received}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, m_s)$ to $\mathfrak{P}_j$ and ignore further *Receive*-message with the same ssid from $\mathfrak{P}_j$.

Figure 7.3: Ideal Functionality for 1-out-of-n Oblivious Transfer $\boldsymbol{\mathcal{F}_{(1,n)\text{-OT}}}$

[ABB+13] was the first step, we considered once again the Haralambiev commitment as the basis of our construction, we used it to do a bit-per-bit commitment to line, and then used several SPHF to hide every line behind noise besides the expected one.

[BC15] was a side-step. We aimed at generalizing Haralambiev commitment, and we did so by explaining it as combination of a CCA-2 encryption with (explainable/verifiable) Chameleon Hashes. We achieved two things with this: on one hand we generalized our construction to non elliptic curve hypotheses, on the other we managed to get rid of the pairing and rely on the classical DDH problem. It should be noted that this is done by abusing from the required artificial pre-flow, and *probably* if in the future, we manage to get rid of this flow, then the technique proposed specifically for DDH will not be transposable.

[BC16] introduced Structure-Preserving Hash Functions, which for PAKE this allowed to commit to the line in only one constant size commitment. The server still needs to send elements for every line, however the user now has a constant communication cost. However, this requires pairing...

[BCG17] proposed a generic transformation from PAKE to Oblivious-Transfer (the other way was

already known). We managed to reexplain various existing protocols, and also build a new Oblivious Transfer from the most efficient PAKE to date [JR15]. While not directly based on SPHF, this Oblivious Transfer is the most efficient to date, and uses similar tricks[2].

The table 7.1 presents a comparison of those various schemes with preexisting ones. As usual, it should be noted that that some optimizations arise when setting $n = 2$.

Table 7.1: Comparison to existing Oblivious Transfer

Paper	Assumption	# Group elements	# Rounds
Static Security (1-out-of-2)			
[PVW08] + [GWZ09]	SXDH	51	8
[CKWZ13]	SXDH	$26 + 7\ \mathbb{Z}_p$	4
Adaptive Security (1-out-of-2)			
[ABB+13]	SXDH	$12\ \mathbb{G}_1 + 1\ \mathbb{G}_2 + 2\mathbb{Z}_p$	3
[BC15]	DDH	$15\ \mathbb{G} + 2\ \mathbb{Z}_p$	3
[BC16]	SXDH	$12\ \mathbb{G}_1 + 4\ \mathbb{G}_2 + 2\ \mathbb{Z}_p$	3
[BCG17]	SXDH	$6\ \mathbb{G}_1 + 2\ \mathbb{G}_2 + 2\ \mathbb{Z}_p$	3
Adaptive Security (1-out-of-n)			
[ABB+13]	SXDH	$\log n\ \mathbb{G}_1 + (n + 8\log n)\ \mathbb{G}_2 + n\ \mathbb{Z}_p$	3
[BC15]	DDH	$(n + 9\log n + 4)\ \mathbb{G} + 2n\ \mathbb{Z}_p$	3
[BC16]	SXDH	$4\ \mathbb{G}_1 + (4n + 4)\ \mathbb{G}_2 + n\ \mathbb{Z}_p$	3
[BCG17]	SXDH	$4\ \mathbb{G}_1 + (n + 2)\ \mathbb{G}_2 + n\ \mathbb{Z}_p$	3

7.3 Adaptive Oblivious Transfer

Due to their huge interest in practice, it is important to achieve low communication on these Oblivious Transfer protocols. A usual drawback is that the server usually has to send a message equivalent to the whole database each time the user requests a line. If it is logical, in the UC framework, that an OT protocol requires a cost linear in the size of the database for the first line queried[3]. One may then hope to amortize the cost for further queries between the same server and the same user (or even another user, if possible), reducing the efficiency gap between Private Information Retrieval schemes and their stronger equivalent Oblivious Transfer schemes. We thus deal in this paper with a more efficient way, which is to achieve *Adaptive* Oblivious Transfer, in which the user can adaptively ask several lines of the database. In such schemes, the server only sends his database once at the beginning of the protocol, and all the subsequent communication is in $o(n)$, more precisely logarithmic. The linear cost is batched once and for all in this preprocessing phase, achieving then a logarithmic complexity closer to the best PIR schemes.

Adaptive versions of those protocols were already studied by [NP97,GH07,KNP11], and more recently UC secure instantiations were proposed, but unfortunately either under the Random Oracle, or under not so standard assumptions such as q-Hidden LRSW or later on q-SDH [CNs07, JL09a, RKP09, CDH12, GD14], but without allowing adaptive corruptions.

We tried to improve this in two ways, first we allowed adaptive corruptions on the user side. The server has to commit the database at the very beginning, hence it seems complicated to allow corruptions beyond this point[4], while relying on a standard hypothesis (k-MDDH).

7.3.1 Transformation

Our construction builds upon the UC-secure OT scheme from [BC15], with ideas inspired from [GH07], who propose a neat framework allowing to achieve *adaptive* Oblivious Transfer (but not in the UC framework). Their construction is quite simple: It requires a *blind Identity-Based Encryption*, in other words, an IBE scheme in which there is a way to query for a user key generation without the authority (here the server) learning the targeted identity (here the line in the database). Once such a Blind IBE

[2] It seems likely that QA-NIZK and SPHF are very close, however due to the argument that SPHF can not exist for all NP, a separation will remain

[3] Assuming adaptive corruptions, we need to be able to explain the last flow once corrupted, hence the last flow needs enough entropy to be equivocated back to the full database.

[4] With our approach one would *simply* need a non-committing affine identity-based encryption.

The functionality $\mathcal{F}_{\mathrm{OT}}^{\mathscr{L}}$ is parametrized by a security parameter $\mathfrak{K}$ and a set of languages $(\mathscr{L}_1, \ldots, \mathscr{L}_n)$ along with the corresponding public verification algorithms $(\mathsf{Verif}_1, \ldots, \mathsf{Verif}_n)$. It interacts with an adversary $\mathcal{S}$ and a set of parties $\mathfrak{P}_1, \ldots, \mathfrak{P}_N$ via the following queries:

- **Upon receiving an input $(\boldsymbol{NewDataBase}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1, \ldots, m_n))$ from party $\mathfrak{P}_i$** , with $m_i \in \{0,1\}^{\mathfrak{K}}$ for all i: record the tuple $(\mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1, \ldots, m_n))$ and reveal $(\mathit{Send}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j)$ to the adversary $\mathcal{S}$. Ignore further *NewDataBase*-message with the same ssid from $\mathfrak{P}_i$.
- **Upon receiving an input $(\boldsymbol{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, \chi_i)$ from party $\mathfrak{P}_j$**: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1, \ldots, m_n))$ is not recorded. Otherwise, reveal $(\mathit{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j)$ to the adversary $\mathcal{S}$ and send $(\mathit{Received}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, m_i')$ to $\mathfrak{P}_j$ where $m_i' = m_i$ if $\mathsf{Verif}_i(\chi_i, \mathscr{L}_i)$ returns 1, and $m_i' = \perp$ otherwise.
  (Non-Adaptive case: Ignore further Receive-message with the same ssid *from $\mathfrak{P}_j$.)*

Figure 7.4: Ideal Functionality for (Adaptive) Oblivious Transfer $\boldsymbol{\mathcal{F}}_{\mathrm{OT}}^{\mathscr{L}}$

is defined, one can conveniently obtain an oblivious transfer protocol by asking the database to encrypt (once and for all) each line for an identity (the j-th line being encrypted for the identity j), and having the user do a blind user key generation query for identity i in order to recover the key corresponding to the line i he expects to learn.

This approach is round-optimal: After the database preparation, the first flow is sent by the user as a commitment to the identity i, and the second one is sent by the server with the blinded requested information. But several technicalities arise because of the UC framework we consider here. For instance, the blinded expected information has to be masked, we do this here using an SPHF. Furthermore, instead of using simple line numbers as identities, we have to commit to words in specific languages (so as to ensure extractability and equivocability) as well as to *fragment* the IBE keys into bits in order to achieve $O(\log n)$ in both flows. This allows us to achieve the first UC-secure adaptive OT protocol allowing adaptive corruptions of the user.

7.3.2 Constructing a Blind Fragmented IBKEM from an IBKEM

Definition and Security Properties of a Blind IBKEM Scheme.

We follow the definition and security definitions used in [BKP14] for an Identity-Based Key Encapsulation (IBKEM) scheme. We continue to follow the KEM formalism by adapting the definition of a Blind IBE scheme given in [GH07] to this setting.

Blind Identity-Based Key Encapsulation Scheme
⌜ A BlindIBKEM consists of four PPT algorithms $(\mathsf{Gen}, \mathsf{BlindUSKGen}, \mathsf{Enc}, \mathsf{Dec})$ with the following properties:

- Gen, Enc and Dec are defined as for a traditional IBKEM scheme.
- $\mathsf{BlindUSKGen}(\langle(\mathcal{S}, \mathsf{msk})(\mathcal{U}, \mathsf{id}, \ell; \boldsymbol{\rho})\rangle)$ is an interactive protocol, in which an honest user $\mathcal{U}$ with identity $\mathsf{id} \in \mathsf{ID}$ obtains the corresponding user secret key $\mathsf{usk}[\mathsf{id}]$ from the master authority $\mathcal{S}$ or outputs an error message, while $\mathcal{S}$'s output is nothing or an error message (ℓ is a label and $\boldsymbol{\rho}$ the randomness).

⌟

Defining the security of a BlindIBKEM requires two additional properties, stated as follows (see [GH07, pages 6 and 7] for the formal security games):

1. **Leak-free Secret Key Generation** (called Leak-free Extract for Blind IBE security in the original paper): A potentially malicious user cannot learn anything by executing the BlindUSKGen protocol with an honest authority which he could not have learned by executing the USKGen protocol with an honest authority; Moreover, as in USKGen, the user must know the identity for which he is extracting a key.

2. **Selective-failure Blindness**: A potentially malicious authority cannot learn anything about the user's choice of identity during the BlindUSKGen protocol; moreover, the authority cannot cause the BlindUSKGen protocol to fail in a manner dependent on the user's choice.

For our applications, we only need a weakened property for blindness:[5]

3. **Weak Blindness**: A potentially malicious authority cannot learn anything about the user's choice of identity during the BlindUSKGen protocol.

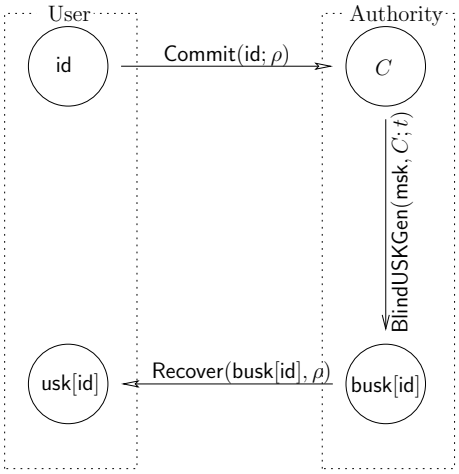
High-Level Idea of the Transformation.

We now show how to obtain a BlindIBKEM scheme from any IBKEM scheme. From a high-level point of view, this transformation mixes two pre-existing approaches.

First, we are going to consider a reverse Naor transform [BF01, CFH+07]: He drew a parallel between Identity-Based Encryption schemes and signature schemes, by showing that a user secret key on an identity can be viewed as the signature on this identity, the verification process therefore being a test that any chosen valid ciphertext for the said identity can indeed be decrypted using the *signature* scheme.

Then, we are going to use Fischlin [Fis06] round-optimal approach to blind signatures, where the whole interaction is done in one pass: First, the user commits to the message, then he recovers a signature linked to his commitment. For sake of simplicity, instead of using a Non-Interactive Zero-Knowledge Proof of Knowledge of a signature, we are going to follow the [BFPV10, BPV12] approach, where thanks to an additional term, the user can extract a signature on the identity from a signature on the committed identity.

Omitting technical details described more precisely in the following sections, the main idea of the transformation of the IBKEM scheme in order to blind a user key request is described in Figure 7.5, page 49.



1. A user commits to the targeted identity id using some randomness ρ.
2. The authority possesses an algorithm allowing it to generate keys for committed identities using its master secret key msk, and some randomness t, in order to obtain a blinded user secret key busk[id].
3. The user using solely the randomness used in the initial commitment is able to recover the requested secret key from the authority's generated value.

Figure 7.5: Generic Transformation of an IBKEM into a Blind IBKEM (naive approach)

Generic Transformation of an IBKEM into a Blind IBKEM.

It now remains to explain how one can fulfill the idea highlighted in Figure 7.5, page 49. The technique to blind a user key request uses a smooth projective hash function, and is often called *implicit decommitment* in recent works: the IBKEM secret key is sent hidden in such a way that it can only be recovered if the user knows how to open the initial commitment on the correct identity. We assume the existence of a labeled CCA-encryption scheme compatible with an SPHF. By "compatible", we mean that the SPHF can be defined over a language $\mathscr{L}_{C,\mathsf{id}} \subset X$, where $\mathscr{L}_{C,\mathsf{id}} = \{C \mid \exists \boldsymbol{\rho} \text{ such that } C = \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}(\mathsf{id}; \boldsymbol{\rho})\}$. We additionally use a key derivation function KDF to derive a pseudo-random bit-string $K \in \{0,1\}^{\mathfrak{K}}$ from a pseudo-random element v. One can use the Leftover-Hash Lemma [HILL99], with a random seed defined in param during the global setup, to extract the entropy from v, then followed by a pseudo-random generator to get a long enough bit-string. Many uses of the same seed in the Leftover-Hash Lemma

[5] Two things to note: First, Selective Failure would be considered as a Denial of Service in the Oblivious Transfer setting. Then, we do not restrict ourselves to schemes where the blindness adversary has access to the generated user keys, as reliable erasures in the OT protocol provide us a way to forget them before being corrupted (otherwise we would need to use a randomizable base IBE).

just lead to a security loss linear in the number of extractions. This gives the following protocol for BlindUSKGen, described in Figure 7.6, page 50.

- The user computes an encryption of the expected identity id and keeps the randomness ρ: $C = \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}(\mathsf{id}; \boldsymbol{\rho})\}$.
- For every identity id', the server computes $\mathsf{usk}[\mathsf{id}']$ along with a pair of (secret, public) hash keys $(\mathsf{hk}_{\mathsf{id}'}, \mathsf{hp}_{\mathsf{id}'})$ for a smooth projective hash function on the language $\mathscr{L}_{C,\mathsf{id}'}$: $\mathsf{hk}_{\mathsf{id}'} = \mathsf{HashKG}(\ell, \mathscr{L}_{C,\mathsf{id}'}, \mathsf{param})$ and $\mathsf{hp}_{\mathsf{id}'} = \mathsf{ProjKG}(\mathsf{hk}_{\mathsf{id}'}, \ell, (\mathscr{L}_{C,\mathsf{id}'}, \mathsf{param}))$. He also compute the corresponding hash value $H_{\mathsf{id}'} = \mathsf{Hash}(\mathsf{hk}_{\mathsf{id}'}, (\mathscr{L}_{C,\mathsf{id}'}, \mathsf{param}), (\ell, C))$. Finally, he sends $(\mathsf{hp}_{\mathsf{id}'}, \mathsf{usk}[\mathsf{id}'] \oplus \mathtt{KDF}(H_{\mathsf{id}'}))$ for every id', where $\oplus$ is a compatible operation.
- Thanks to $\mathsf{hp}_{\mathsf{id}}$, the user is able to compute the corresponding projected hash value $H'_{\mathsf{id}} = \mathsf{ProjHash}(\mathsf{hp}_{\mathsf{id}}, (\mathscr{L}_{C,\mathsf{id}}, \mathsf{param}), (\ell, C), \boldsymbol{\rho})$. He then recovers $\mathsf{usk}[\mathsf{id}]$ for the initially committed identity id since $H_{\mathsf{id}} = H'_{\mathsf{id}}$.

Figure 7.6: Summary of the Generic Construction of $\mathsf{BlindUSKGen}(\langle(\mathcal{S}, \mathsf{msk})(\mathcal{U}, \mathsf{id}, \ell; \boldsymbol{\rho})\rangle)$ for a blind IBE

Theorem 7.3.1 *If* IBKEM *is a* PR-ID-CPA*-secure identity-based key encapsulation scheme and $\mathcal{E}$ a labeled* CCA*-encryption scheme compatible with an* SPHF*, then* BlindIBKEM *is leak free and weak blind.*

Proof: First, BlindIBKEM satisfies leak-free secret key generation since it relies on the CCA security on the encryption scheme, forbidding a user to open it to another identity than the one initially encrypted. Furthermore, the pseudo-randomness of the SPHF ensures that the blinded user key received for id is indistinguishable from random if he encrypted $\mathsf{id}' \neq \mathsf{id}$. Finally, the weak blindness also relies on the CCA security on the encryption scheme, since an encryption of id is indistinguishable from a encryption of $\mathsf{id}' \neq \mathsf{id}$. □

Using a Blind IBKEM in our Application to Adaptive Oblivious Transfer.

The previous approach allows to transform an IBKEM into a Blind IBKEM, but it has a huge drawback in our context: Since we assume an exponential identity space, it requires an exponential number of answers from the authority, which cannot help us to fulfill logarithmic complexity in our application. However, if we focus on the special case of affine IBE with bitwise function[6], a user key can be described as the list $(\mathsf{usk}[0], \mathsf{usk}[0, \mathsf{id}_0], \ldots, \mathsf{usk}[m-1, \mathsf{id}_{m-1}])$ if id_i is the i-th bit of the identity id. One can thus manage to be much more efficient by sending each "bit" evaluation on the user secret key, hidden with a smooth projective hash value on the language "the i-th bit of the identity is a 0 (or 1)", which is common to all identities. We can thus reduce the number of languages from the number of identities (which is exponential) to the length of an identity (which is polynomial). For security reasons, one cannot give directly the evaluation value, but as we are considering the sum of the evaluations for each bit, we simply add a Shamir-like secret sharing, by adding randomness that is going to cancel out at the end.

As a last step, we finally need to make our construction compatible with the UC framework with adaptive corruptions. In this context, interactions should make sense for any possible input chosen by the environment and learnt a posteriori in the simulation during the corruption of an honest party. From the user side, this implies that the last flow should contain enough recoverable information so that a simulator, having sent a commitment to an incorrect identity, can extract the proper user secret key corresponding to the correct identity recovered after the corruption. From the server side, this implies that the IBKEM scheme is defined such as one is able to adapt the user secret keys in order to correspond to the new database learnt a posteriori. Of course, not all schemes allow this property, but this will be the case in the pairing scenario considered in our concrete instantiation.

To deal with corruptions of the user, recall that a simulated server (knowing the secret key of the encryption scheme) is already able to extract the identity committed to. But we now consider that, for all id, $\mathscr{L}_{\mathsf{id}}$ is the language of the equivocable commitments on words in the inner language $\widetilde{\mathscr{L}_{\mathsf{id}}} = \{\mathsf{id}\}$. We assume them to be a *Trapdoor Collection of Languages*, which means that it is computationally hard to sample an element in $\mathscr{L}_1 \cap \cdots \cap \mathscr{L}_n$, except for the simulator, who possesses a trapdoor tk (the equivocation trapdoor) allowing it to sample an element in the intersection of languages. This allows a simulated user (knowing this trapdoor) not to really bind to any identity during the commitment phase. The only difference with the algorithm described in Figure 7.7, page 51 is that the user now encrypts

[6] They were defined in [BKP14]. Affine IBE derive their name from the fact that only affine operations are done on the identity bits (no hashing, square rooting, inverting... are allowed).

- The user computes a bit-per-bit encryption of the expected identity id and keeps the randomness ρ: $C = \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}(\mathsf{id}; \boldsymbol{\rho})\}$.
- The server computes a fragmented version of all the keys $\mathsf{usk}[\mathsf{id}']$, i.e. all the values $\mathsf{usk}[i, b]$ for i from 0 up to the length m of the keys and $b \in \{0, 1\}$. He also computes a pair of (secret, public) hash keys $(\mathsf{hk}_{i,b}, \mathsf{hp}_{i,b})$ for a smooth projective hash function on the language $\mathscr{L}_{C,i,b}$: "The i-th bit of value encrypted into C is b", i.e. $\mathsf{hk}_{i,b} = \mathsf{HashKG}(\ell, \mathscr{L}_{C,i,b}, \mathsf{param})$ and $\mathsf{hp}_{i,b} = \mathsf{ProjKG}(\mathsf{hk}_{i,b}, \ell, (\mathscr{L}_{C,i,b}, \mathsf{param}))$. He also computes the corresponding hash value $H_{i,b} = \mathsf{Hash}(\mathsf{hk}_{i,b}, (\mathscr{L}_{C,i,b}, \mathsf{param}), (\ell, C))$ and chooses random values z_i. Finally, he sends, for each (i, b), $(\mathsf{hp}_{i,b}, \mathsf{busk}[i, b])$, where $\mathsf{busk}[i, b] = \mathsf{usk}[i, b] \oplus \mathtt{KDF}(H_{i,b}) \oplus z_i$, together with $Z = \mathsf{usk}_0 \ominus \left(\bigoplus_i z_i\right)$, where $\oplus$ is a compatible operation and $\ominus$ its inverse.
- Thanks to the $\mathsf{hp}_{i,\mathsf{id}_i}$ for the initially committed identity id, the user is able to compute the corresponding projected hash value $H'_{i,\mathsf{id}_i} = \mathsf{ProjHash}(\mathsf{hp}_{i,\mathsf{id}_i}, (\mathscr{L}_{C,i,\mathsf{id}_i}, \mathsf{param}), (\ell, C), \boldsymbol{\rho})$, that should be equal to H_{i,id_i} for all i. From the values $\mathsf{busk}[i, \mathsf{id}_i]$, he then recovers $\mathsf{usk}[i, \mathsf{id}_i] \oplus z_i$. Finally, with the operation $\left(\bigoplus_i (\mathsf{usk}[i, \mathsf{id}_i] \oplus z_i)\right) \oplus Z$, he recovers the expected $\mathsf{usk}[\mathsf{id}]$.

Figure 7.7: Summary of the Generic Construction of $\mathsf{BlindUSKGen}(\langle(\mathcal{S}, \mathsf{msk})(\mathcal{U}, \mathsf{id}, \ell; \boldsymbol{\rho})\rangle)$ for a Blind affine IBE

this word χ (which is an equivocable commitment on his identity id) rather than directly encrypting his identity id: $C = \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}(\chi; \boldsymbol{\rho})$.

7.3.3 Generic Construction of Adaptive OT

We derive from here our generic construction of OT (depicted in Figure 7.8, page 52). We additionally assume the existence of a Pseudo-Random Generator (PRG) F with input size equal to the plaintext size, and output size equal to the size of the messages in the database and an $\mathsf{IND-CPA}$encryption scheme $\mathcal{E} = (\mathsf{Setup}_{\mathrm{cpa}}, \mathsf{KeyGen}_{\mathrm{cpa}}, \mathsf{Encrypt}_{\mathrm{cpa}}, \mathsf{Decrypt}_{\mathrm{cpa}})$ with plaintext size at least equal to the security parameter. First, the owner of the database generates the keys for such an IBE scheme, and encrypts each line i of the database for the identity i. Then when a user wants to request a given line, he runs the blind user key generation algorithm and recovers the key for the expected given line. This leads to the following security result, proven in [BCG16].

Theorem 7.3.2 *Assuming that* $\mathsf{BlindUSKGen}$ *is constructed as described above, the adaptive Oblivious Transfer protocol described in Figure 7.8, page 52 UC-realizes the functionality* $\mathcal{F}^{\mathscr{L}}_{OT}$ *presented in Figure 7.4, page 48 with adaptive corruptions assuming reliable erasures.*

7.3.4 Pairing-Based Instantiation of Adaptive OT

Affine Bit-Wise Blind IBE. In [BKP14], we proposed a generic framework to move from affine Message Authentication Code to IBE, and then a tight instantiation of such a MAC, giving an affine bit-wise IBE, which seems like a good candidate for our setting (making it blind and fragmented).

We are thus going to use the family of IBE described in the following picture (Figure 7.9, page 52), which is their instantiation derived from a Naor-Reingold MAC[7]. In the following, $h_i()$ are injective deterministic public functions mapping a bit to a scalar in $\mathbb{Z}_p$.

A property that was not studied in this paper was the blind user key generation: How to generate and answer blind user secret key queries? We answer to this question by proposing the k-MDDH-based variation presented in Figure 7.10, page 53. To fit the global framework we are going to consider the language of Haralambiev commitments. We denote this process as Har in the following protocol, and by $\mathscr{L}_{\mathsf{Har},i,\mathsf{id}_i}$ the language on identity bits. We thus obtain the following security results.

Theorem 7.3.3 *This construction achieves both the weak Blindness, and the leak-free secret key generation requirements under the k-MDDH assumption.*

The first one is true under the indistinguishability of the generalized Cramer-Shoup encryption, as the server learns nothing about the line requested during the first flow. It should even be noted that because of the inner chameleon hash, a simulator is able to use the trapdoor to do a commitment to

[7] For the reader familiar with the original result, we combine $x, \mathbf{y}$ into a bigger $\mathbf{y}$ to lighten the notations, and compact the (x'_i, y'_i) values into a single y' as this has no impact on their construction.

CRS generation:
$\mathsf{crs} \xleftarrow{\$} \mathsf{SetupCom}(1^{\mathfrak{K}}), \mathsf{param}_{\mathrm{cpa}} \xleftarrow{\$} \mathsf{Setup}_{\mathrm{cpa}}(1^{\mathfrak{K}})$.

Database Preparation:
1. Server runs $\mathsf{Gen}(\mathfrak{K})$, to obtain $\mathsf{mpk}, \mathsf{msk}$.
2. For each line t, he computes $(D_t, K_t) = \mathsf{Enc}(\mathsf{mpk}, t)$, and $L_t = K_t \oplus DB(t)$.
3. He also computes $\mathsf{usk}[i, b]$ for all $i = 1 \ldots, m$ and $b = 0, 1$ and erases msk.
4. Server generates a key pair $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathrm{cpa}}(\mathsf{param}_{\mathrm{cpa}})$ for $\mathcal{E}$, stores sk and completely erases the random coins used by KeyGen.
5. He then publishes $\mathsf{mpk}, \{(D_t, L_t)\}_t, \mathsf{pk}$.

Index query on s:
1. User chooses a random value S, computes $R \leftarrow F(S)$ and encrypts S under pk:
   $c \xleftarrow{\$} \mathsf{Encrypt}_{\mathrm{cpa}}(\mathsf{pk}, S)$
2. User computes $\mathcal{C}$ with the first flow of $\mathsf{BlindUSKGen}(\langle(\mathcal{S}, \mathsf{msk})(\mathcal{U}, s, \ell; \boldsymbol{\rho})\rangle)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathcal{U}, \mathcal{S})$ (see Figure 7.7, page 51).
3. User stores the random $\boldsymbol{\rho}_s = \{\boldsymbol{\rho}_*\}$ needed to open $\mathcal{C}$ to s, and completely erases the rest, including the random coins used by $\mathsf{Encrypt}_{\mathrm{cpa}}$ and sends $(c, \mathcal{C})$ to the Server

IBE input msk:
1. Server decrypts $S \leftarrow \mathsf{Decrypt}_{\mathrm{cpa}}(\mathsf{sk}, c)$ and computes $R \leftarrow F(S)$
2. Server runs the second flow of $\mathsf{BlindUSKGen}(\langle(\mathcal{S}, \mathsf{msk})(\mathcal{U}, s, \ell; \boldsymbol{\rho})\rangle)$ on $\mathcal{C}$ (see Figure 7.7, page 51).
3. Server erases every new value except $(\mathsf{hp}_{i,b})_{i,b}, (\mathsf{busk}[i, b])_{i,b}, Z \oplus R$ and sends them over a secure channel.

Data recovery:
1. User then using, $\boldsymbol{\rho}_s$ recovers $\mathsf{usk}[s]$ from the values received from the server.
2. He can then recover the expected information with $\mathsf{Dec}(\mathsf{usk}[s], s, D_s) \oplus L_s$ and erases everything else.

Figure 7.8: Adaptive UC-Secure 1-out-of-n OT from a Fragmented Blind IBE

$\underline{\mathsf{Gen}(\mathfrak{K})}$:
$\mathbf{A} \xleftarrow{\$} \mathcal{D}_k, \mathbf{B} = \overline{\mathbf{A}}$
For $i \in [\![0, \ell]\!] : \mathbf{Y}_i \xleftarrow{\$} \mathbb{Z}_p^{k+1}; \mathbf{Z}_i = \mathbf{Y}_i^\top \cdot \mathbf{A} \in \mathbb{Z}_p^k$
$\mathbf{y}' \xleftarrow{\$} \mathbb{Z}_p^{k+1}; \mathbf{z}' = \mathbf{y}'^\top \cdot \mathbf{A} \in \mathbb{Z}_p^k$
$\mathsf{mpk} := (\mathcal{G}, [\mathbf{A}]_1, ([\mathbf{Z}_i]_1)_{i \in [\![0,\ell]\!]}, [\mathbf{z}']_1)$
$\mathsf{msk} := (\mathbf{Y}_i)_{i \in [\![0,\ell]\!]}, \mathbf{y}'$
Return $(\mathsf{mpk}, \mathsf{msk})$

$\underline{\mathsf{USKGen}(\mathsf{msk}, \mathsf{id})}$:
$\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^k, \mathbf{t} = \mathbf{B}\mathbf{s}$
$\mathbf{w} = (\mathbf{Y}_0 \sum_{i=1}^{\ell} \mathsf{id}_i \mathbf{Y}_i)\mathbf{t} + \mathbf{y}' \in \mathbb{Z}_p^{k+1}$
Return $\mathsf{usk}[\mathsf{id}] := ([\mathbf{t}]_2, [\mathbf{w}]_2) \in \mathbb{G}_2^{k+k+1}$

$\underline{\mathsf{Enc}(\mathsf{mpk}, \mathsf{id})}$:
$\mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^k$
$\mathbf{c}_0 = \mathbf{A}\mathbf{r} \in \mathbb{Z}_p^{k+1}$
$\mathbf{c}_1 = (\mathbf{Z}_0 \sum_{i=0}^{\ell} h_i(\mathsf{id}_i)\mathbf{Z}_i) \cdot \mathbf{r} \in \mathbb{Z}_p$
$K = \mathbf{z}' \cdot \mathbf{r} \in \mathbb{Z}_p$.
Return $[K]_T$ and $\mathsf{C} = ([\mathbf{c}_0]_1, [\mathbf{c}_1]_1) \in \mathbb{G}_1^{k+1+1}$

$\underline{\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, \mathsf{C})}$:
Parse $\mathsf{usk}[\mathsf{id}] = ([\mathbf{t}]_2, [\mathbf{w}]_2)$
Parse $\mathsf{C} = ([\mathbf{c}_0]_1, [\mathbf{c}_1]_1)$
$K = e([\mathbf{c}_0]_1, [\mathbf{w}]_2) \cdot e([\mathbf{c}_1]_1, [\mathbf{t}]_2)^{-1}$
Return $K \in \mathbb{G}_T$

Figure 7.9: A fragmentable affine IBKEM.

every possible words of the set of languages at once, and so can adaptively decide which id he requested. The proof of the second result is detailed in [BCG16].

For sake of generality, any bit-wise affine IBE could work (like for example Waters IBE [Wat05]), the additional price paid for tightness here is very small and allows to have a better reduction in the proof, but it it not required by the framework itself.

Adaptive UC-Secure Oblivious Transfer. We finally get our instantiation by combining this k-MDDH-based blind IBE with a $k - \mathsf{MDDH}$ variant of ElGamal for the CPA encryption needed. The requirement on the IBE blind user secret key generation (being able to adapt the key if the line changes) is achieved assuming that the server knows the discrete logarithms of the database lines. This is quite easy to achieve by assuming that for all line s, $DB(s) = [db(s)]_1$ where $db(s)$ is the real line (thus known). It

- First flow: $\mathcal{U}$ starts by computing
  $\boldsymbol{\rho} \xleftarrow{\$} \mathbb{Z}_p^{1+4\times\ell}$,
  $\mathbf{a}, \mathbf{d} = \mathsf{Har}(\mathsf{id}, \ell; \rho) \in \mathbb{Z}_p^{\ell} \times \mathbb{Z}_p^{2\times(k+3)\ell}$,
  Sends $\mathcal{C} = ([\mathbf{a}]_1, [\mathbf{d}]_2)$ to S
- Second Flow: $\mathcal{S}$ then proceeds
  $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^k$, $\mathbf{t} = \mathbf{B}\mathbf{s}$, $\mathbf{f} \xleftarrow{\$} \mathbb{Z}_p^{\ell\times k+1}$,
  For each $i \in [\![1, \lceil \log n \rceil]\!], b \in [\![0, 1]\!]$:
  $\mathsf{hk}_{i,b} = \mathsf{HashKG}(\mathscr{L}_{\mathsf{Har},i,b}, \mathbf{C})$
  $\mathsf{hp}_{i,b} = \mathsf{ProjKG}(\mathsf{hk}_{i,b}, \mathscr{L}_{\mathsf{Har},i,b}, \mathbf{C})$
  $H_{i,b} = \mathsf{Hash}(\mathsf{hk}_{i,b}, \mathscr{L}_{\mathsf{Har},i,b}, \mathbf{C})$
  $\omega_{i,b} = (b\mathbf{Y}_i)\mathbf{t} + \mathbf{f}_i + H_{i,b}$

Then sets $\mathbf{w}_0 = \mathbf{Y}_0\mathbf{t} + \mathbf{y}' - \sum_{i=1}^{\ell} \mathbf{f}_i \in \mathbb{Z}_p^{k+1}$
Returns $\mathsf{busk} :=$
$([\mathbf{t}]_2, [\mathbf{w}_0]_2, \{[\boldsymbol{\omega}_{i,b}]_2\}, \{[\mathsf{hp}_{i,b}]_2\})$

- $\mathsf{BlindUSKGen}_3$: $\mathcal{U}$ then recovers his key
  For each $i \in [\![1, \ell]\!]$:
  $H'_i =$
  $\mathsf{ProjHash}(\mathsf{hp}_{i,\mathsf{id}_i}, \mathscr{L}_{\mathsf{Har},i,\mathsf{id}_i}, \mathbf{C}, \rho_i)$
  $\mathbf{w}_i = \omega_{i,\mathsf{id}_i} - H'_i$
  $\mathbf{w} = \mathbf{w}_0 + \sum_{i=1}^{\ell} \mathbf{w}_i$
  And then recovers $\mathsf{usk}[\mathsf{id}] :=$
  $[\mathbf{t}]_2, [\mathbf{w}]_2$

Figure 7.10: $\mathsf{BlindUSKGen}(\langle(\mathcal{S}, \mathsf{msk})(\mathcal{U}, \mathsf{id}, \ell; \boldsymbol{\rho})\rangle)$.

implies a few more computation on the user's side in order to recover $db(s)$ from $DB(s)$, but this remains completely feasible if the lines belong to a small space. For practical applications, one could imagine to split all 256-bit lines into 8 pieces for a decent/constant trade-off in favor of computational efficiency.

For $k = 1$, so under the classical SXDH assumption, the first flow requires $8\log|DB|$ elements in $\mathbb{G}_1$ for the CCA encryption part and $\log(|DB|+1)$ in $\mathbb{G}_2$ for the chameleon one, while the second flow would now require $1 + 4\log|DB|$ elements in $\mathbb{G}_1$, $1 + 2\log|DB|$ for the fragmented masked key, and $2\log|DB|$ for the projection keys.

7.4 Oblivious Signature-Based Envelope

Oblivious Signature-Based Envelope (OSBE) were introduced in [LDB03]. It can be viewed as an efficient way to ease the asymmetrical aspect of several authentication protocols. Alice is a member of an organization and possesses a certificate produced by an authority attesting she is in this organization. Bob wants to send a private message P to members of this organization. However due to the sensitive nature of the organization, Alice does not want to give Bob neither her certificate nor a proof she belongs to the organization. OSBE lets Bob sends an obfuscated version of this message P to Alice, in such a way that Alice will be able to find P if and only if Alice is in the required organization. In the process, Bob cannot decide whether Alice does really belong to the organization.

Oblivious Signature-Based Envelope
⌜ An OSBE scheme is defined by four algorithms ($\mathsf{OSBESetup}, \mathsf{OSBEKeyGen}, \mathsf{OSBESign}, \mathsf{OSBEVerif}$), and one interactive protocol $\mathsf{OSBEProtocol}\langle\mathcal{S}, \mathcal{R}\rangle$:

- $\mathsf{OSBESetup}(1^k)$, where k is the security parameter, generates the global parameters param;
- $\mathsf{OSBEKeyGen}(\mathsf{param})$ generates the keys $(\mathsf{vk}, \mathsf{sk})$ of the certification authority;
- $\mathsf{OSBESign}(\mathsf{sk}, m)$ produces a signature σ on the input message m, under the signing key sk;
- $\mathsf{OSBEVerif}(\mathsf{vk}, m, \sigma)$ checks whether σ is a valid signature on m, w.r.t. the public key vk; it outputs 1 if the signature is valid, and 0 otherwise.
- $\mathsf{OSBEProtocol}\langle\mathcal{S}(\mathsf{vk}, M, P), \mathcal{R}(\mathsf{vk}, M, \sigma)\rangle$ between the sender $\mathcal{S}$ with the private message P, and the recipient $\mathcal{R}$ with a certificate σ. If σ is a valid signature under vk on the common message M, then $\mathcal{R}$ receives P, otherwise it receives nothing. In any case, $\mathcal{S}$ does not learn anything.

⌟

Such an OSBE scheme should be (the three last properties are additional —or stronger— security properties from the original definitions [LDB03]):

- *correct*: the protocol actually allows $\mathcal{R}$ to learn P, whenever σ is a valid signature on M under vk;
- *oblivious*: the sender should not be able to distinguish whether $\mathcal{R}$ uses a valid signature σ on M under vk as input. More precisely, if $\mathcal{R}_0$ knows and uses a valid signature σ and $\mathcal{R}_1$ does not use such a valid signature, the sender cannot distinguish an interaction with $\mathcal{R}_0$ from an interaction with $\mathcal{R}_1$;
- *(weakly) semantically secure*: the recipient learns nothing about $\mathcal{S}$ input P if it does not use a valid signature σ on M under vk as input. More precisely, if $\mathcal{S}_0$ owns P_0 and $\mathcal{S}_1$ owns P_1, the recipient that does not use a valid signature cannot distinguish an interaction with $\mathcal{S}_0$ from an interaction with $\mathcal{S}_1$;
- *semantically secure* (denoted sem): the above indistinguishability should hold even if the receiver has seen several interactions $\langle\mathcal{S}(\mathsf{vk}, M, P), \mathcal{R}(\mathsf{vk}, M, \sigma)\rangle$ with valid signatures, and the same sender's input P;

$\mathsf{Exp}^{\mathsf{obl}_A-b}_{\mathcal{OSBE},\mathcal{A}}(k)$ [Escrow Free property]
1. $\mathsf{param} \leftarrow \mathsf{OSBESetup}(1^k)$
2. $\mathsf{vk} \leftarrow \mathcal{A}(\texttt{INIT} : \mathsf{param})$
3. $(M, \sigma) \leftarrow \mathcal{A}(\texttt{FIND} : \texttt{Send}(\mathsf{vk}, \cdot, \cdot), \texttt{Rec}^*(\mathsf{vk}, \cdot, \cdot, 0), \texttt{Exec}^*(\mathsf{vk}, \cdot, \cdot, \cdot))$
4. $\mathsf{OSBEProtocol}\langle \mathcal{A}, \texttt{Rec}^*(\mathsf{vk}, M, \sigma, b)\rangle$
5. $b' \leftarrow \mathcal{A}(\texttt{GUESS} : \texttt{Send}(\mathsf{vk}, \cdot, \cdot), \texttt{Rec}^*(\mathsf{vk}, \cdot, \cdot, 0), \texttt{Exec}^*(\mathsf{vk}, \cdot, \cdot, \cdot))$
6. `RETURN` b'

$\mathsf{Exp}^{\mathsf{sem}^*-b}_{\mathcal{OSBE},\mathcal{A}}(k)$ [Semantic security w.r.t. the authority]
1. $\mathsf{param} \leftarrow \mathsf{OSBESetup}(1^k)$
2. $\mathsf{vk} \leftarrow \mathcal{A}(\texttt{INIT} : \mathsf{param})$
3. $(M, \sigma, P_0, P_1) \leftarrow \mathcal{A}(\texttt{FIND} : \texttt{Send}(\mathsf{vk}, \cdot, \cdot), \texttt{Rec}^*(\mathsf{vk}, \cdot, \cdot, 0), \texttt{Exec}^*(\mathsf{vk}, \cdot, \cdot, \cdot))$
4. $\mathsf{transcript} \leftarrow \mathsf{OSBEProtocol}\langle \texttt{Send}(\mathsf{vk}, M, P_b), \texttt{Rec}^*(\mathsf{vk}, M, \sigma, 0\rangle$
5. $b' \leftarrow \mathcal{A}(\texttt{GUESS} : \mathsf{transcript}, \texttt{Send}(\mathsf{vk}, \cdot, \cdot), \texttt{Rec}^*(\mathsf{vk}, \cdot, \cdot, 0), \texttt{Exec}^*(\mathsf{vk}, \cdot, \cdot, \cdot))$
6. `RETURN` b'

$\mathsf{Exp}^{\mathsf{sem}-b}_{\mathcal{OSBE},\mathcal{A}}(k)$ [Semantic Security]
1. $\mathsf{param} \leftarrow \mathsf{OSBESetup}(1^k)$
2. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{OSBEKeyGen}(\mathsf{param})$
3. $(M, P_0, P_1) \leftarrow \mathcal{A}(\texttt{FIND} : \mathsf{vk}, \texttt{Sign}^*(\mathsf{vk}, \cdot), \texttt{Send}(\mathsf{vk}, \cdot, \cdot), \texttt{Rec}(\mathsf{vk}, \cdot, 0), \texttt{Exec}(\mathsf{vk}, \cdot, \cdot))$
4. $\mathsf{OSBEProtocol}\langle \texttt{Send}(\mathsf{vk}, M, P_b), \mathcal{A}\rangle$
5. $b' \leftarrow \mathcal{A}(\texttt{GUESS} : \texttt{Sign}(\mathsf{vk}, \cdot), \texttt{Send}(\mathsf{vk}, \cdot, \cdot), \texttt{Rec}(\mathsf{vk}, \cdot, 0), \texttt{Exec}(\mathsf{vk}, \cdot, \cdot))$
6. `IF` $M \in \mathcal{SM}$ `RETURN` 0 `ELSE RETURN` b'

Figure 7.11: Security Games for $\mathcal{OSBE}$

- *escrow free* (denoted obl_A): the authority (owner of the signing key sk), playing as the sender or just eavesdropping, is unable to distinguish whether $\mathcal{R}$ used a valid signature σ on M under vk as input. This notion supersedes the above *oblivious* property, since this is essentially oblivious w.r.t. the authority, without any restriction.
- *semantically secure w.r.t. the authority* (denoted sem^*): after the interaction, the authority (owner of the signing key sk) learns nothing about P.

We insist that the escrow-free property (obl_A) is stronger than the oblivious property, hence we will consider the former only. However, the semantic security w.r.t. the authority (sem^*) is independent from the basic semantic security (sem) since in the latter the adversary interacts with the sender whereas in the former the adversary (who generated the signing keys) has only passive access to a challenge transcript.

These security notions can be formalized by the security games presented on Figure 7.11, page 54, where the adversary keeps some internal state between the various calls `INIT`, `FIND` and `GUESS`. They make use of the oracles described below, and the advantages of the adversary are, for all the security notions,

$$\mathsf{Adv}^*_{\mathcal{OSBE},\mathcal{A}}(k) = \Pr[\mathsf{Exp}^{*-1}_{\mathcal{OSBE},\mathcal{A}}(k) = 1] - \Pr[\mathsf{Exp}^{*-0}_{\mathcal{OSBE},\mathcal{A}}(k) = 1]$$

$$\mathsf{Adv}^*_{\mathcal{OSBE}}(k, t) = \max_{\mathcal{A} \le t} \mathsf{Adv}^*_{\mathcal{OSBE},\mathcal{A}}(k).$$

- $\texttt{Sign}(\mathsf{vk}, m)$: This oracle outputs a valid signature on m under the signing key sk associated to vk (where the pair $(\mathsf{vk}, \mathsf{sk})$ has been outputted by the $\mathsf{OSBEKeyGen}$ algorithm);
- $\texttt{Sign}^*(\mathsf{vk}, m)$: This oracle first queries $\texttt{Sign}(\mathsf{vk}, m)$. It additionally stores the query m to the list $\mathcal{SM}$;
- $\texttt{Send}(\mathsf{vk}, m, P)$: This oracle emulates the sender with private input P, and thus may consist of multiple interactions;
- $\texttt{Rec}(\mathsf{vk}, m, b)$: This oracle emulates the recipient either with a valid signature σ on m under the verification key vk (obtained from the signing oracle $\texttt{Sign}$) if $b = 0$ (as the above $\mathcal{R}_0$), or with a random string if $b = 1$ (as the above $\mathcal{R}_1$). This oracle is available when the signing key has been generated by $\mathsf{OSBEKeyGen}$ only;
- $\texttt{Rec}^*(\mathsf{vk}, m, \sigma, b)$: This oracle does as above, with a valid signature σ provided by the adversary. If $b = 0$, it emulates the recipient playing with σ; if $b = 1$, it emulates the recipient playing with a random string;
- $\texttt{Exec}(\mathsf{vk}, m, P)$: This oracle outputs the transcript of an honest execution between a sender with private input P and the recipient with a valid signature σ on m under the verification key vk

(obtained from the signing oracle $\texttt{Sign}$). It basically activates the $\texttt{Send}(\mathsf{vk}, m, P)$ and $\texttt{Rec}(\mathsf{vk}, m, 0)$ oracles.

- $\texttt{Exec}^*(\mathsf{vk}, m, \sigma, P)$: This oracle outputs the transcript of an honest execution between a sender with private input P and the recipient with a valid signature σ (provided by the adversary). It basically activates the $\texttt{Send}(\mathsf{vk}, m, P)$ and $\texttt{Rec}^*(\mathsf{vk}, m, \sigma, 0)$ oracles.

Remark The OSBE schemes proposed in [LDB03] do not satisfy the semantic security w.r.t. the authority. This is obvious for the generic construction based on identity-based encryption which consists in only one flow of communication (since a scheme that achieves the strong security notions requires at least two flows). This is also true (to a lesser extent) for the RSA-based construction: for any third party, the semantic security relies (in the random oracle model) on the CDH assumption in a 2048-bit RSA group; but for the authority, it can be broken by solving two 1024-bit discrete logarithm problems. This task is much simpler in particular if the authority generates the RSA modulus $N = pq$ dishonestly (*e.g.* with $p-1$ and $q-1$ smooth). In order to make the scheme secure in our strong model, one needs (at least) to double the size of the RSA modulus and to make sure that the authority has selected and correctly employed a truly random seed in the generation of the RSA key pair [JG02].

7.4.1 High-Level Instantiation

We assume we have an encryption scheme $\mathcal{E}$, a signature scheme $\mathcal{S}$ and an SPHF system onto a set $\mathbb{G}$. We additionally use a key derivation function KDF to derive a pseudo-random bit-string $K \in \{0,1\}^\ell$ from a pseudo-random element v in $\mathbb{G}$. One can use the Leftover-Hash Lemma [HILL99], with a random seed defined in param during the global setup, to extract the entropy from v, then followed by a pseudo-random generator to get a long enough bit-string. Many uses of the same seed in the Leftover-Hash-Lemma just leads to a security loss linear in the number of extractions. We describe an oblivious signature-based envelope system $\mathcal{OSBE}$, to send a private message $P \in \{0,1\}^\ell$:

- $\mathsf{OSBESetup}(1^k)$, where k is the security parameter:
  - it first generates the global parameters for the signature scheme , the encryption scheme, and the SPHF system (using Setup);
  - it then generates the public key ek of the encryption scheme (using $\mathsf{KeyGen}_{\mathcal{E}}$, while the decryption key will not be used);

  The output param consists of all the individual param and the encryption key ek;
- $\mathsf{OSBEKeyGen}(\mathsf{param})$ runs $\mathsf{KeyGen}_{\mathcal{S}}(\mathsf{param})$ to generate a pair $(\mathsf{vk}, \mathsf{sk})$ of verification-signing keys;
- The $\mathsf{OSBESign}$ and $\mathsf{OSBEVerif}$ algorithms are exactly Sign and Verif from the signature scheme;
- $\mathsf{OSBEProtocol}\langle\mathcal{S}(\mathsf{vk}, M, P), \mathcal{R}(\mathsf{vk}, M, \sigma)\rangle$: In the following, $\mathscr{L} = \mathscr{L}(\mathsf{vk}, M)$ will describe the language of the ciphertexts under the above encryption key ek of a valid signature of the input message M under the input verification key vk (hence vk and M as inputs, while param contains ek).
  - $\mathcal{R}$ generates and sends $c = \mathsf{Encrypt}(\mathsf{ek}, \sigma; r)$;
  - $\mathcal{S}$ computes $\mathsf{hk} = \mathsf{HashKG}(\mathscr{L}, \mathsf{param})$, $\mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathscr{L}, \mathsf{param}), c)$, $v = \mathsf{Hash}(\mathsf{hk}, (\mathscr{L}, \mathsf{param}), c)$, and $Q = P \oplus \mathsf{KDF}(v)$; $\mathcal{S}$ sends hp, Q to $\mathcal{R}$;
  - $\mathcal{R}$ computes $v' = \mathsf{ProjHash}(\mathsf{hp}, (\mathscr{L}, \mathsf{param}), c, r)$ and $P' = Q \oplus \mathsf{KDF}(v')$.

One can do much more under this global framework. We can handle priced / conditional oblivious transfer, where we (obliviously) checked if the user is allowed to access a line because his credentials are valid or if its bank account has enough money.

One can also consider it, to design universal designated verifier signature like we did in [BCGJ17], where the verified learns whether someone signed a message but is then unable to prove it to someone else.

The same way, one can formalize Blind Signatures in this framework [BPV12], where the *Server/Signer* allows to sign the message if and only if the user is able to prove that the value he committed in the first flow is indeed belonging to the set of valid messages.[8]

$\star$

[8] In fact, the construction we used to build the blind IBE Key Retrieval is a blind signature

Conclusion

While working on concrete protocols and their UC instantiation, we managed to formalize two big families of constructions, and generic constructions associated with them. This allowed us then to focus on improving the inner building blocks obtaining very efficient protocols while doing so.

In this section, we are now going to recall the main results, we obtained and more importantly (part of) what remains to be done.

At the core of our protocols, we rely on Smooth Projective Hash Functions. In the past years, we manage to tackle several challenges, like building KV-SPHF for CCA-2 encryption, providing SPHF for post-quantum hypotheses, or even building SPHF for non-membership statement. We also extended existing constructions to allow the use of a group element as a witness.

This very last result has already allowed us to build *compact* schemes, but one could still explore what is available now that we can combine NIZK and SPHF, because this now introduce some flexibility that Smoothness was preventing. For example, one can try to apply this to the recent NIKE result from [HHK18] based on Hash Proofs.

Languages Another direction, that might be really interesting to study, is to further precise the border of things manageable with an SPHF, we know we can not handle all NP, but we can handle affine languages, techniques on lattices/codes highlight that we can handle neighborhood of affine languages, and our result also show that we can handle compliment. Can we do more?

An intermediate result would also be to see, what languages can be handled by KV SPHF, it seems unlikely that all languages manageable with an SPHF can be KV, but where should we stop.

A last challenge would be to obtain a full collection on post-quantum candidates, and manage to design an SPHF around the SIDH problem.

QA-NIZK As of now, UC protocols seem to be more efficient when instantiated with QA-NIZK. As SPHF seem to require less powerful cryptographic techniques, it would seem likely that this is not credible, either we should be instantiate protocol that can not be handled via QA-NIZK, or for a given thing, we should have a more efficient SPHF.

This is somewhat already the case, as we can avoid using pairings on elliptic curves, but there is still something there.

In all likelihood, this is going to arrive via simultaneous evaluation. For now, when we build a PAKE (for example), each user checks that the other's password is what they expect, and proves that its password is valid. One might expect to build multi-SPHF, where both users work together to prove the password is equal mutualizing the overhead caused by the SPHF.

Distrusting the world Every scheme we propose relies on setup assumptions, perfect randomness, and so on. One important lines of research would be to see what would happen in real life? What happens if the CRS is corrupted? We have impossibility results on PAKE and Angel-based cryptography, but what happens if just a part of the CRS should be distrusted? Or now, if we assume we have a proper, proper standards, what happens if during the hash key generation we have a diminished entropy?

Bibliography

[ABB+13] Michel Abdalla, Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, and David Pointcheval. SPHF-friendly non-interactive commitments. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 214–234, Bengalore, India, December 1–5, 2013. Springer, Heidelberg, Germany. 4, 28, 29, 36, 37, 38, 45, 46, 47

[ABCG15] Quentin Alamélou, Olivier Blazy, Stéphane Cauchie, and Philippe Gaborit. A Code-Based Group Signature Scheme. In Jean-Pierre Tillich Pascale Charpin, Nicolas Sendrier, editor, *The 9th International Workshop on Coding and Cryptography 2015 WCC2015*, Proceedings of the 9th International Workshop on Coding and Cryptography 2015 WCC2015, Paris, France, April 2015. 4

[ABCG16] Quentin Alamélou, Olivier Blazy, Stéphane Cauchie, and Philippe Gaborit. A practical group signature scheme based on rank metric. In Sylvain Duquesne and Svetla Petkova-Nikova, editors, *Arithmetic of Finite Fields - 6th International Workshop, WAIFI 2016, Ghent, Belgium, July 13-15, 2016, Revised Selected Papers*, volume 10064 of *Lecture Notes in Computer Science*, pages 258–275. Springer, Heidelberg, Germany, July 2016. 4

[ABCG17] Quentin Alamélou, Olivier Blazy, Stéphane Cauchie, and Philippe Gaborit. A code-based group signature scheme. *Designs, Codes and Cryptography*, 82:1–25, 2017. 1, 4

[ACFP05] Michel Abdalla, Olivier Chevassut, Pierre-Alain Fouque, and David Pointcheval. A simple threshold authenticated key exchange from short secrets. In Bimal K. Roy, editor, *Advances in Cryptology – ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 566–584, Chennai, India, December 4–8, 2005. Springer, Heidelberg, Germany. 39, 40

[ACP09] Michel Abdalla, Céline Chevalier, and David Pointcheval. Smooth projective hashing for conditionally extractable commitments. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 671–689, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Heidelberg, Germany. 13, 14, 27, 28, 29, 33, 34, 35, 36, 38

[AFG+10] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 209–236, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany. 20

[AIR01] William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 119–135, Innsbruck, Austria, May 6–10, 2001. Springer, Heidelberg, Germany. 2

[AKB07] Giuseppe Ateniese, Jonathan Kirsch, and Marina Blanton. Secret handshakes with dynamic and fuzzy matching. In *ISOC Network and Distributed System Security Symposium – NDSS 2007*, San Diego, CA, USA, February 28 – March 2, 2007. The Internet Society. 1, 42

[Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *44th Annual Symposium on Foundations of Computer Science*, pages 298–307, Cambridge, MA, USA, October 11–14, 2003. IEEE Computer Society Press. 3

[AMBD+18] Carlos Aguilar Melchor, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, and Gilles Zémor. Efficient encryption from random quasi-cyclic codes. *IEEE Trans. Information Theory*, 64(5):3927–3943, 2018. 1, 3, 4, 24

[Att14] Nuttapong Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 557–577, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany. 2

[BBC+13a] Fabrice Ben Hamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. Efficient UC-secure authenticated key-exchange for algebraic languages. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013: 16th International Conference on Theory and Practice of Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 272–291, Nara, Japan, February 26 – March 1, 2013. Springer, Heidelberg, Germany. 4, 19, 28, 33, 35

[BBC+13b] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 449–475, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany. 2, 14, 15, 33, 36, 37, 38

[BBDQ18] Fabrice Benhamouda, Olivier Blazy, Léo Ducas, and Willy Quach. Hash proof systems over lattices revisited. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 10770 of *Lecture Notes in Computer Science*, pages 644–674, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Heidelberg, Germany. 4, 21

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany. 30

[BC15] Olivier Blazy and Céline Chevalier. Generic construction of UC-secure oblivious transfer. In Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis, editors, *ACNS 15: 13th International Conference on Applied Cryptography and Network Security*, volume 9092 of *Lecture Notes in Computer Science*, pages 65–86, New York, NY, USA, June 2–5, 2015. Springer, Heidelberg, Germany. 4, 28, 29, 45, 46, 47

[BC16] Olivier Blazy and Céline Chevalier. Structure-preserving smooth projective hashing. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part II*, volume 10032 of *Lecture Notes in Computer Science*, pages 339–369, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany. 3, 18, 20, 21, 31, 37, 38, 45, 46, 47

[BCG16] Olivier Blazy, Céline Chevalier, and Paul Germouty. Adaptive oblivious transfer and generalization. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part II*, volume 10032 of *Lecture Notes in Computer Science*, pages 217–247, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany. 2, 3, 43, 51, 52

[BCG17] Olivier Blazy, Céline Chevalier, and Paul Germouty. Almost optimal oblivious transfer from QA-NIZK. In Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi, editors, *ACNS 17: 15th International Conference on Applied Cryptography and Network Security*, volume 10355 of *Lecture Notes in Computer Science*, pages 579–598, Kanazawa, Japan, July 10–12, 2017. Springer, Heidelberg, Germany. 4, 45, 46, 47

[BCGJ17] Olivier Blazy, Emmanuel Conchon, Paul Germouty, and Amandine Jambert. Efficient id-based designated verifier signature. In *Proceedings of the 12th International Conference on Availability, Reliability and Security, Reggio Calabria, Italy, August 29 - September 01, 2017*, pages 44:1–44:8. ACM, 2017. 55

[BCL+05] Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin. Secure computation without authentication. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 361–377, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Heidelberg, Germany. 35, 36

[BCPV13] Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. Analysis and improvement of Lindell's UC-secure commitment schemes. In Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *ACNS 13: 11th International Conference on Applied Cryptography and Network Security*, volume 7954 of *Lecture Notes in Computer Science*, pages 534–551, Banff, AB, Canada, June 25–28, 2013. Springer, Heidelberg, Germany. 4, 27

[BCV15] Olivier Blazy, Céline Chevalier, and Damien Vergnaud. Non-interactive zero-knowledge proofs of non-membership. In Kaisa Nyberg, editor, *Topics in Cryptology – CT-RSA 2015*, volume 9048 of *Lecture Notes in Computer Science*, pages 145–164, San Francisco, CA, USA, April 20–24, 2015. Springer, Heidelberg, Germany. 17, 25

[BCV16] Olivier Blazy, Céline Chevalier, and Damien Vergnaud. Mitigating server breaches in password-based authentication: Secure and efficient solutions. In Kazue Sako, editor, *Topics in Cryptology – CT-RSA 2016*, volume 9610 of *Lecture Notes in Computer Science*, pages 3–18, San Francisco, CA, USA, February 29 – March 4, 2016. Springer, Heidelberg, Germany. 38

[BDS+03] Dirk Balfanz, Glenn Durfee, Narendar Shankar, Diana K. Smetters, Jessica Staddon, and Hao-Chi Wong. Secret handshakes from pairing-based key agreements. In *IEEE Symposium on Security and Privacy*, pages 180–196. IEEE Computer Society, 2003. 1, 33, 42

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany. 49

[BFI+10] Olivier Blazy, Georg Fuchsbauer, Malika Izabachène, Amandine Jambert, Hervé Sibert, and Damien Vergnaud. Batch Groth-Sahai. In Jianying Zhou and Moti Yung, editors, *ACNS 10: 8th International Conference on Applied Cryptography and Network Security*, volume 6123 of *Lecture Notes in Computer Science*, pages 218–235, Beijing, China, June 22–25, 2010. Springer, Heidelberg, Germany. 1

[BFPV10] Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Signatures on randomizable ciphertexts. In Rosario Gennaro, editor, *Proceedings of PKC 2011*, Lecture Notes in Computer Science. Springer, 2010. Full version available from the web page of the authors. 49

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Joe Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341, Cambridge, MA, USA, February 10–12, 2005. Springer, Heidelberg, Germany. 2

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 12: 19th Conference on Computer and Communications Security*, pages 784–796, Raleigh, NC, USA, October 16–18, 2012. ACM Press. 2

[BJKS03] John G. Brainard, Ari Juels, Burt Kaliski, and Michael Szydlo. A new two-server approach for authentication with short secrets. In *Proceedings of the 12th USENIX Security Symposium, Washington, D.C., USA, August 4-8, 2003*, 2003. 39

[BKKP15] Olivier Blazy, Saqib A. Kakvi, Eike Kiltz, and Jiaxin Pan. Tightly-secure signatures from chameleon hash functions. In Jonathan Katz, editor, *PKC 2015: 18th International Conference on Theory and Practice of Public Key Cryptography*, volume 9020 of *Lecture Notes in Computer Science*, pages 256–279, Gaithersburg, MD, USA, March 30 – April 1, 2015. Springer, Heidelberg, Germany. 1, 4

[BKP14] Olivier Blazy, Eike Kiltz, and Jiaxin Pan. (Hierarchical) identity-based encryption from affine message authentication. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 408–425, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany. 1, 3, 4, 48, 50, 51

[BM92] Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *1992 IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, May 1992. 1, 33, 36

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155, Bruges, Belgium, May 14–18, 2000. Springer, Heidelberg, Germany. 38, 40

[BPV12] Olivier Blazy, David Pointcheval, and Damien Vergnaud. Round-optimal privacy-preserving protocols with smooth projective hash functions. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 94–111, Taormina, Sicily, Italy, March 19–21, 2012. Springer, Heidelberg, Germany. 49, 55

[BW06] Xavier Boyen and Brent Waters. Compact group signatures without random oracles. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 427–444, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany. 1

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press. 8, 9, 36, 39

[CCGS10] Jan Camenisch, Nathalie Casati, Thomas Groß, and Victor Shoup. Credential authenticated identification and key exchange. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 255–276, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany. 2, 33

[CCL15] Ran Canetti, Asaf Cohen, and Yehuda Lindell. A simpler variant of universally composable security for standard multiparty computation. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 3–22, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany. 9

[CDH12] Jan Camenisch, Maria Dubovitskaya, and Kristiyan Haralambiev. Efficient structure-preserving signature scheme from standard assumptions. In Ivan Visconti and Roberto De Prisco, editors, *SCN 12: 8th International Conference on Security in Communication Networks*, volume 7485 of *Lecture Notes in Computer Science*, pages 76–94, Amalfi, Italy, September 5–7, 2012. Springer, Heidelberg, Germany. 47

[CDN09] Jan Camenisch, Maria Dubovitskaya, and Gregory Neven. Oblivious transfer with access control. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *ACM CCS 09: 16th Conference on Computer and Communications Security*, pages 131–140, Chicago, Illinois, USA, November 9–13, 2009. ACM Press. 2

[CDNZ11] Jan Camenisch, Maria Dubovitskaya, Gregory Neven, and Gregory M. Zaverucha. Oblivious transfer with hidden access control policies. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011: 14th International Conference on Theory and Practice of Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 192–209, Taormina, Italy, March 6–9, 2011. Springer, Heidelberg, Germany. 2

[CEN15] Jan Camenisch, Robert R. Enderlein, and Gregory Neven. Two-server password-authenticated secret sharing UC-secure against transient corruptions. In Jonathan Katz, editor, *PKC 2015: 18th International Conference on Theory and Practice of Public Key*

Cryptography, volume 9020 of *Lecture Notes in Computer Science*, pages 283–307, Gaithersburg, MD, USA, March 30 – April 1, 2015. Springer, Heidelberg, Germany. 39

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 19–40, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany. 27, 36

[CFH+07] Yang Cui, Eiichiro Fujisaki, Goichiro Hanaoka, Hideki Imai, and Rui Zhang. Formal security treatments for signatures from identity-based encryption. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *ProvSec 2007: 1st International Conference on Provable Security*, volume 4784 of *Lecture Notes in Computer Science*, pages 218–227, Wollongong, Australia, November 1–2, 2007. Springer, Heidelberg, Germany. 49

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th Annual Symposium on Foundations of Computer Science*, pages 41–50, Milwaukee, Wisconsin, October 23–25, 1995. IEEE Computer Society Press. 2

[CHK+05] Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally composable password-based key exchange. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 404–421, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany. 34, 35, 36

[CK02] Ran Canetti and Hugo Krawczyk. Universally composable notions of key exchange and secure channels. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 337–351, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer, Heidelberg, Germany. 36

[CKWZ13] Seung Geol Choi, Jonathan Katz, Hoeteck Wee, and Hong-Sheng Zhou. Efficient, adaptively secure, and composable oblivious transfer with a single, global CRS. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013: 16th International Conference on Theory and Practice of Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 73–88, Nara, Japan, February 26 – March 1, 2013. Springer, Heidelberg, Germany. 47

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th Annual ACM Symposium on Theory of Computing*, pages 494–503, Montréal, Québec, Canada, May 19–21, 2002. ACM Press. 27

[CNs07] Jan Camenisch, Gregory Neven, and abhi shelat. Simulatable adaptive oblivious transfer. In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 573–590, Barcelona, Spain, May 20–24, 2007. Springer, Heidelberg, Germany. 47

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25, Santa Barbara, CA, USA, August 23–27, 1998. Springer, Heidelberg, Germany. 33

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer, Heidelberg, Germany. 13, 15

[CS03] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 126–144, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany. 27

[DG03] Mario Di Raimondo and Rosario Gennaro. Provably secure threshold password-authenticated key exchange. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 507–523, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany. 39

[DG06] Mario Di Raimondo and Rosario Gennaro. Provably secure threshold password-authenticated key exchange. *J. Comput. Syst. Sci.*, 72(6):978–1001, 2006. 39

[DG17] Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 537–569, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. 4

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. 10

[DN02] Ivan Damgård and Jesper Buus Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 581–596, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Heidelberg, Germany. 27

[DOR99] Giovanni Di Crescenzo, Rafail Ostrovsky, and Sivaramakrishnan Rajagopalan. Conditional oblivious transfer and timed-release encryption. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 74–89, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg, Germany. 2

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 129–147, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany. 18, 30

[Fis06] Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 60–77, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Heidelberg, Germany. 49

[FK00] Warwick Ford and Burton S. Kaliski Jr. Server-assisted generation of a strong secret from a password. In *9th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2000), 4-16 June 2000, Gaithersburg, MD, USA*, pages 176–180, 2000. 38, 39

[FLM11] Marc Fischlin, Benoît Libert, and Mark Manulis. Non-interactive and re-usable universally composable string commitments with adaptive security. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 468–485, Seoul, South Korea, December 4–8, 2011. Springer, Heidelberg, Germany. 3, 27, 29, 30, 38

[FPV09] Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Transferable constant-size fair e-cash. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *CANS 09: 8th International Conference on Cryptology and Network Security*, volume 5888 of *Lecture Notes in Computer Science*, pages 226–247, Kanazawa, Japan, December 12–14, 2009. Springer, Heidelberg, Germany. 1

[GD14] Vandana Guleria and Ratna Dutta. Lightweight universally composable adaptive oblivious transfer. In ManHo Au, Barbara Carminati, and C.-C.Jay Kuo, editors, *Network and System Security*, volume 8792 of *Lecture Notes in Computer Science*, pages 285–298. Springer International Publishing, 2014. 47

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 467–476, Palo Alto, CA, USA, June 1–4, 2013. ACM Press. 17

[GH07] Matthew Green and Susan Hohenberger. Blind identity-based encryption and simulatable oblivious transfer. In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 265–282, Kuching, Malaysia, December 2–6, 2007. Springer, Heidelberg, Germany. 47, 48

[GIKM98] Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting data privacy in private information retrieval schemes. In *30th Annual ACM Symposium on Theory of Computing*, pages 151–160, Dallas, TX, USA, May 23–26, 1998. ACM Press. 2

[GKW15] Romain Gay, Iordanis Kerenidis, and Hoeteck Wee. Communication complexity of conditional disclosure of secrets and attribute-based encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 485–502, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany. 2

[GL03] Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 524–543, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany. `http://eprint.iacr.org/2003/032.ps.gz`. 13, 14, 36, 39

[GL06] Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. *ACM Transactions on Information and System Security*, 9(2):181–234, 2006. 33

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. 9

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988. 10

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. 1

[GOS18] Sanjam Garg, Rafail Ostrovsky, and Akshayaram Srinivasan. Adaptive garbled RAM from laconic oblivious transfer. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 515–544, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany. 2

[Gro07] Jens Groth. Fully anonymous group signatures without random oracles. In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 164–180, Kuching, Malaysia, December 2–6, 2007. Springer, Heidelberg, Germany. 1

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany. 19

[GWZ09] Juan A. Garay, Daniel Wichs, and Hong-Sheng Zhou. Somewhat non-committing encryption and efficient adaptively secure oblivious transfer. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 505–523, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Heidelberg, Germany. 47

[Har11] Kristiyan Haralambiev. *Efficient Cryptographic Primitives for Non-Interactive Zero-Knowledge Proofs and Applications*. PhD thesis, New York University, 2011. 27, 28

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 341–371, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany. 24

[HHK18] Julia Hesse, Dennis Hofheinz, and Lisa Kohl. On tightly secure non-interactive key exchange. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 65–94, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany. 56

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999. 45, 49, 55

[HJKY95] Amir Herzberg, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive secret sharing or: How to cope with perpetual leakage. In Don Coppersmith, editor, *Advances in Cryptology – CRYPTO'95*, volume 963 of *Lecture Notes in Computer Science*, pages 339–352, Santa Barbara, CA, USA, August 27–31, 1995. Springer, Heidelberg, Germany. 40

[HK12] Shai Halevi and Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. *Journal of Cryptology*, 25(1):158–193, January 2012. 33

[HMQ04] Dennis Hofheinz and Jörn Müller-Quade. Universally composable commitments using random oracles. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 58–76, Cambridge, MA, USA, February 19–21, 2004. Springer, Heidelberg, Germany. 30

[IW14] Yuval Ishai and Hoeteck Wee. Partial garbling schemes and their applications. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *ICALP 2014: 41st International Colloquium on Automata, Languages and Programming, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 650–662, Copenhagen, Denmark, July 8–11, 2014. Springer, Heidelberg, Germany. 2

[Jab01] David P. Jablon. Password authentication using multiple servers. In David Naccache, editor, *Topics in Cryptology – CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 344–360, San Francisco, CA, USA, April 8–12, 2001. Springer, Heidelberg, Germany. 39

[JG02] Ari Juels and Jorge Guajardo. RSA key generation with verifiable randomness. In David Naccache and Pascal Paillier, editors, *PKC 2002: 5th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2274 of *Lecture Notes in Computer Science*, pages 357–374, Paris, France, February 12–14, 2002. Springer, Heidelberg, Germany. 55

[JL09a] Stanislaw Jarecki and Xiaomin Liu. Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 577–594. Springer, Heidelberg, Germany, March 15–17, 2009. 47

[JL09b] Stanislaw Jarecki and Xiaomin Liu. Private mutual authentication and conditional oblivious transfer. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 90–107, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Heidelberg, Germany. 1, 42

[JR12] Charanjit S. Jutla and Arnab Roy. Relatively-sound NIZKs and password-based key-exchange. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012: 15th International Conference on Theory and Practice of Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 485–503, Darmstadt, Germany, May 21–23, 2012. Springer, Heidelberg, Germany. 33

[JR13] Charanjit S. Jutla and Arnab Roy. Shorter quasi-adaptive NIZK proofs for linear subspaces. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 1–20, Bengalore, India, December 1–5, 2013. Springer, Heidelberg, Germany. 1

[JR15] Charanjit S. Jutla and Arnab Roy. Dual-system simulation-soundness with applications to UC-PAKE and more. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015, Part I*, volume 9452 of *Lecture Notes in Computer Science*, pages 630–655, Auckland, New Zealand, November 30 – December 3, 2015. Springer, Heidelberg, Germany. 3, 36, 38, 47

[Kal05] Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 78–95, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany. 13, 33

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *20th Annual ACM Symposium on Theory of Computing*, pages 20–31, Chicago, IL, USA, May 2–4, 1988. ACM Press. 2

[KM14] Franziskus Kiefer and Mark Manulis. Distributed smooth projective hashing and its application to two-server password authenticated key exchange. In Ioana Boureanu, Philippe Owesarski, and Serge Vaudenay, editors, *ACNS 14: 12th International Conference on Applied Cryptography and Network Security*, volume 8479 of *Lecture Notes in Computer Science*, pages 199–216, Lausanne, Switzerland, June 10–13, 2014. Springer, Heidelberg, Germany. 39

[KMTG05] Jonathan Katz, Philip D. MacKenzie, Gelareh Taban, and Virgil D. Gligor. Two-server password-only authenticated key exchange. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *ACNS 05: 3rd International Conference on Applied Cryptography and Network Security*, volume 3531 of *Lecture Notes in Computer Science*, pages 1–16, New York, NY, USA, June 7–10, 2005. Springer, Heidelberg, Germany. 39

[KMTG12] Jonathan Katz, Philip D. MacKenzie, Gelareh Taban, and Virgil D. Gligor. Two-server password-only authenticated key exchange. *J. Comput. Syst. Sci.*, 78(2):651–669, 2012. 39, 40

[KNP11] Kaoru Kurosawa, Ryo Nojima, and Le Trieu Phong. Generic fully simulatable adaptive oblivious transfer. In Javier Lopez and Gene Tsudik, editors, *ACNS 11: 9th International Conference on Applied Cryptography and Network Security*, volume 6715 of *Lecture Notes in Computer Science*, pages 274–291, Nerja, Spain, June 7–10, 2011. Springer, Heidelberg, Germany. 47

[KOY01] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient password-authenticated key exchange using human-memorable passwords. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 475–494, Innsbruck, Austria, May 6–10, 2001. Springer, Heidelberg, Germany. 36, 39

[KPW15] Eike Kiltz, Jiaxin Pan, and Hoeteck Wee. Structure-preserving signatures from standard assumptions, revisited. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 275–295, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany. 20

[KV09] Jonathan Katz and Vinod Vaikuntanathan. Smooth projective hashing and password-based authenticated key exchange from lattices. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 636–652, Tokyo, Japan, December 6–10, 2009. Springer, Heidelberg, Germany. 4, 21, 22

[KV11] Jonathan Katz and Vinod Vaikuntanathan. Round-optimal password-based authenticated key exchange. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 293–310, Providence, RI, USA, March 28–30, 2011. Springer, Heidelberg, Germany. 2, 15, 21, 33, 36, 38

[KZ09] Aggelos Kiayias and Hong-Sheng Zhou. Zero-knowledge proofs with witness elimination. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009: 12th International Conference on Theory and Practice of Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 124–138, Irvine, CA, USA, March 18–20, 2009. Springer, Heidelberg, Germany. 25

[LDB03] Ninghui Li, Wenliang Du, and Dan Boneh. Oblivious signature-based envelope. In Elizabeth Borowsky and Sergio Rajsbaum, editors, *22nd ACM Symposium Annual on Principles of Distributed Computing*, pages 182–189, Boston, MA, USA, July 13–16, 2003. Association for Computing Machinery. 2, 53, 55

[Lin11] Yehuda Lindell. Highly-efficient universally-composable commitments based on the DDH assumption. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 446–466, Tallinn, Estonia, May 15–19, 2011. Springer, Heidelberg, Germany. 27

[LL07] Sven Laur and Helger Lipmaa. A new protocol for conditional disclosure of secrets and its applications. In Jonathan Katz and Moti Yung, editors, *ACNS 07: 5th International Conference on Applied Cryptography and Network Security*, volume 4521 of *Lecture Notes in Computer Science*, pages 207–225, Zhuhai, China, June 5–8, 2007. Springer, Heidelberg, Germany. 2

[MSJ02] Philip D. MacKenzie, Thomas Shrimpton, and Markus Jakobsson. Threshold password-authenticated key exchange. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 385–400, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Heidelberg, Germany. 39

[NP97] Moni Naor and Benny Pinkas. Visual authentication and identification. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 322–336, Santa Barbara, CA, USA, August 17–21, 1997. Springer, Heidelberg, Germany. 47

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd Annual ACM Symposium on Theory of Computing*, pages 427–437, Baltimore, MD, USA, May 14–16, 1990. ACM Press. 9, 15

[OY91] Rafail Ostrovsky and Moti Yung. How to withstand mobile virus attacks (extended abstract). In Luigi Logrippo, editor, *10th ACM Symposium Annual on Principles of Distributed Computing*, pages 51–59, Montreal, Quebec, Canada, August 19–21, 1991. Association for Computing Machinery. 40

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany. 47

[Rab81] Michael O. Rabin. How to exchange secrets with oblivious transfer. Technical Report TR81, Harvard University, 1981. 2, 43, 45

[RKP09] Alfredo Rial, Markulf Kohlweiss, and Bart Preneel. Universally composable adaptive priced oblivious transfer. In Hovav Shacham and Brent Waters, editors, *PAIRING 2009: 3rd International Conference on Pairing-based Cryptography*, volume 5671 of *Lecture Notes in Computer Science*, pages 231–247, Palo Alto, CA, USA, August 12–14, 2009. Springer, Heidelberg, Germany. 2, 47

[RS92] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444, Santa Barbara, CA, USA, August 11–15, 1992. Springer, Heidelberg, Germany. 9

[SK05] Michael Szydlo and Burton S. Kaliski Jr. Proofs for two-server password authentication. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 227–244, San Francisco, CA, USA, February 14–18, 2005. Springer, Heidelberg, Germany. 39

[SPMLS02] Jacques Stern, David Pointcheval, John Malone-Lee, and Nigel P. Smart. Flaws in applying proof methodologies to signature schemes. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 93–110, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Heidelberg, Germany. 10

[Wat05] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany. 52

[Wee14] Hoeteck Wee. Dual system encryption via predicate encodings. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 616–637, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany. 2

[WHC+14] Xiao Shaun Wang, Yan Huang, T.-H. Hubert Chan, Abhi Shelat, and Elaine Shi. SCORAM: Oblivious RAM for secure computation. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 14: 21st Conference on Computer and Communications Security*, pages 191–202, Scottsdale, AZ, USA, November 3–7, 2014. ACM Press. 2

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 80–91, Chicago, Illinois, November 3–5, 1982. IEEE Computer Society Press. 8

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press. 2

[ZY17] Jiang Zhang and Yu Yu. Two-round PAKE from approximate SPH and instantiations from lattices. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part III*, volume 10626 of *Lecture Notes in Computer Science*, pages 37–67, Hong Kong, China, December 3–7, 2017. Springer, Heidelberg, Germany. 21

Résumé

Dans ce manuscrit, nous revenons sur diverses briques utilisées dans le contexte de la cryptographie implicite. Tout d'abord nous nous concentrons sur les Smooth Projective Hash Functions (*Fonctions de hachage à projections régulières*), nous montrons comment les classifier, comment en construire et comment repousser les limites des langages pouvant être gêrés. Pour celà, nous les étudions tout d'abord sous l'angle classique des courbes elliptiques mais également des réseaux euclidiens, ou même de la cryptographie à base de codes.
Ensuite nous proposons de nouvelles méthodologies pour construire et prouver des protocoles d'échanges de clé authentifiés (que nous regroupons derrière le concept de LAKE : Language Based Authenticated Key Exchange, *Echange de clé authentifié par un langage*), et des protocoles asymétriques (regroupés sous le concept d'OLBE : Oblivious Language-Based Envelope, *Enveloppe Inconsciente basée sur un langage*). A chaque fois, nous fournissions des fonctionnalités idéales, des instantiations génériques et montrons comment instantier les diverses briques pour générer des protocoles sûrs et le plus efficaces possibles. Bien que développées de façon générique, nous remarquons que nos instantiations conduisent à des protocoles extrêmement efficaces même en cas de corruptions adaptatives, et que ces constructions se transposent presque naturellement aux hypothèses post-quantiques.

Abstract

In this manuscript, we consider various building blocks used in the context of implicit cryptography. First, we focus on Smooth Projective Hash Functions, we show how to classify them properly, how to build them, and how to widen the array of languages that can be considered. For that we first start with vanilla cryptography by considering elliptic curves, and then further proceed to post-quantum solutions like euclidean lattices, and code-based cryptography.
Then, we proposed new methodologies to build and prove the security of authenticated key exchanges protocols (we encompass all of them through the notion of LAKE: Language Based Authenticated Key Exchange), and more asymmetric protocols (that we encompass with the notion of OLBE: Oblivious Language-Based Envelope). Each time, we provide specific ideal functionalities, and generic instantiation, and show how to instantiate the various building blocks to achieve efficient and secure protocols. Even if the construction are generic, the protocols remain efficient even in case of adaptive corruptions. Interestingly, due to the generic design, everything can be easily transposed to post-quantum hypotheses.