Zero-Knowledge Proofs with Witness Elimination

Aggelos Kiayias[⋆] and Hong-Sheng Zhou[⋆]

Computer Science and Engineering
University of Connecticut
Storrs, CT, USA
{aggelos,hszhou}@cse.uconn.edu

Abstract. Zero-knowledge proofs with witness elimination are protocols that enable a prover to demonstrate knowledge of a witness to the verifier that accepts the interaction provided that the witness is valid for a given statement and additionally the witness **does not** belong to a set of *eliminated witnesses*. This set is determined by a public relation Q (that parameterizes the primitive) and the *private* input of the verifier. Zero-knowledge proofs with witness elimination thus call for a relaxation of the zero-knowledge property and are relevant in settings where a statement has a multitude of witnesses that may attest to its validity. A number of interesting issues arise in the design of such protocols that include whether a protocol transcript enables the verifier to test for witness after termination (something akin to an "offline dictionary attack") and whether the prover should be capable of understanding whether her witness is eliminated. The primitive is motivated by the setting of identification schemes where a user wishes to authenticate herself to an access point while preserving her anonymity and the access point needs to certify that the user is eligible while at the same time making sure she does not match the identity of a suspect user that is tracked by the authorities. We call such primitives anonymous identification schemes with *suspect tracking*.

In this work we formalize zero-knowledge proofs with witness elimination in the universal composability setting and we provide a general construction based on smooth projective hashing that is suitable for designing efficient schemes. As an illustration of our general construction we then present an explicit efficient scheme for proving knowledge of a Boneh-Boyen signature with witness elimination. Our scheme requires the design of a smooth projective hash function for the language of linear ElGamal ciphertexts. Along the way we demonstrate how zero-knowledge proofs with witness elimination naturally relate to the primitives of password-based key exchange and private equality testing.

[⋆] Research partly supported by NSF CAREER Award CNS-0447808.

1 Introduction

Zero-knowledge proofs were introduced in [23] and constitute a most useful cryptographic primitive. Given the wide applicability of the primitive several generalizations of its basic formulation have been considered, typically by extending the basic set of security properties that are captured by a protocol implementation. These include parallel and concurrent composition of zero-knowledge [22, 18], non-malleable zero-knowledge [17], resettable zero-knowledge protocols [9], monotone closure of zero-knowledge proofs [16], non-interactive zero-knowledge [2] and more recently isolated zero-knowledge proofs [15].

The security of the prover in a zero-knowledge proof is intended to be captured by demonstrating the existence of a simulator that is capable of generating transcripts indistinguishable from regular honest prover verifier interactions. A relaxation of the zero-knowledge property is the notion of witness indistinguishability [19] that requires instead that a malicious verifier cannot distinguish which one amongst two different possible witnesses the prover is using. Note that witness indistinguishability is useful as a notion only in cases where a number of distinct witnesses are associated with the same statement (while in cases when only a single witness exists the property is vacuous).

In this work we consider a different relaxation of the zero-knowledge property that we call *witness elimination.* A zero-knowledge proof with witness elimination with respect to a relation Q, enables the verifier to make sure that the witness employed by the prover does not satisfy $(w, w') \in Q$ where w' is of the choice of the verifier. Our typical example for the relation Q would be the equality relation: in this case, the verifier given a string w' of her choice can make sure that $w \neq w'$. Note that in case w' is public and the relation R for which the proof is executed is in NP, witness elimination can be resolved generically by having the prover engage in a proof that $((x, w) \in R) \wedge (w \neq w')$. Nevertheless witness elimination is interesting as a property in the case that the verifier does not wish to disclose to the prover which witness w' is she checking against.

It follows from the above discussion that zero-knowledge proofs with witness elimination as a primitive introduces a private input for the verifier for which, ideally, no information should be disclosed to the prover about it. A number of interesting questions arise when one seeks to construct protocols that solve the witness elimination problem for zero-knowledge proofs, in particular: (i) how many candidate witnesses should the verifier be able to test given a single protocol execution? (ii) should the verifier be able to test whether the prover possesses a valid witness? (this is relevant in the case where the witness has been eliminated) (iii) should the prover be allowed to extract the information that the verifier is eliminating her witness based on the protocol interaction? (note that the prover may realize this from a signal that is external to the protocol but this may or may not be provided depending on the application scenario – see below).

Our motivation for studying this primitive is its application to the identification protocol setting. Indeed, a notable application of zero-knowledge proofs is the efficient design of identification schemes that was exemplified in the work of Fiat and Shamir [20] and Schnorr [30]. Given such protocols, the ability to perform efficient conjunctions and disjunctions of zero-knowledge proof protocols [16, 12] lead to the design of anonymous identification schemes [6]. Indeed, disjunction of identification protocols leads to a natural one-to-many relationship between a valid statement and the witnesses that attest to its validity something that in turn lends itself to the design of anonymous identification schemes. In particular, a prover may show that he belongs to a group of users provided that such group is identified by a directory of public-keys.

Witness elimination in this domain translates naturally to the following problem: the verifier wishes to make sure that the prover's identity does not match a suspect that is currently being tracked by authorities. Given that anonymity must still be preserved the verifier should have some way to make sure that the witness used by the prover is not equal to a suspect witness while not learning any further information about the witness of the prover. At the same time the protocol should not leak any information about the suspect identity which may be protected for the course of the investigation and potentially should never be revealed if sufficient evidence is not gathered. This is related to but at the same time different from user revocation in identification schemes where typically the identity of revoked users can be made public in the form of a CRL.

Our contributions are summarized as follows:

– We formalize the primitive of zero-knowledge with witness elimination in the Universal Composability setting [7, 8]. In our security formalization we capture the following properties related to the three questions raised above: (i) A verifier can only test a single witness for each live protocol execution with the prover; this also ensures the prover that transcripts of her protocol executions at the present time cannot be tracked if she becomes a suspect in the future. This protects the privacy of individuals prior to the time when authorities receive a court order that enables their tracking. As a cryptographic property it also relates to protection against *offline dictionary attacks* in password based authenticated key exchange [25, 21, 10] as well as to the notion of *backwards unlinkability* of identification schemes [31, 1, 27] in the context of user revocation. (ii) In our protocols we may allow the verifier to have the power to test whether the witness of the prover is valid even in case that such witness is eliminated – this is natural as we view zero-knowledge proofs with witness elimination as an extension of the standard zero-knowledge proof functionality. (iii) We do not allow any information about the input of the verifier to be leaked to the prover from a protocol execution, i.e., the prover will be oblivious to what witnesses are eliminated by the verifier (independently of whether she is revoked or not). This protects the identity of the suspect in case no case is ever made against her by the authorities.

– We provide a general construction of zero-knowledge proofs with witness elimination that is suitable for building efficient implementations and is based on the notion of smooth projective hash functions, [13, 21, 24].
– We present an explicit practical construction of a zero-knowledge proof with witness elimination for the language of all non-adaptive Boneh-Boyen signatures [3]. This result immediately implies an anonymous identification scheme with suspect tracking. As part of our construction we also design a smooth projective hash function family for the language of Linear ElGamal ciphertexts that is suitable in the bilinear group setting when the DDH assumption may not necessary hold.

Comparison with previous primitives. We note first that the problem of zero-knowledge with witness elimination can be viewed within the general context of secure two party computation (as it is also the case for regular zero-knowledge proofs). This means that one can apply generic techniques to obtain a solution for the functionality (e.g., using the protocol compiler of [11]). Still such generic techniques do not yield efficient protocols or much insight about the primitive something that further motivates the present investigation. Second, depending on the relation Q, zero-knowledge proofs with witness elimination may provide an inequality test for the private inputs of the prover and the verifier. It follows that this makes the resulting primitive similar to private equality tests [28] and password-based key exchange [25, 21, 10]. Note though that in the latter primitive it is also required to incorporate a private coin flipping component (that will produce the shared key) something that is not necessary in the witness elimination setting. On the other hand, the property that protocol transcripts of a password-based key exchange should not facilitate offline dictionary attacks corresponds to the property that the verifier should not be able to extract any new information from old witness elimination transcripts.

Looking at the primitive from the identification point of view, one can draw parallels to the primitive of traceable signatures [26] and verifier-local revocation group signatures [31, 1, 5]. The difference of these previous schemes compared to anonymous identification with suspect tracking is that in our case prior identification transcripts need to conceal the identity of the users even after the suspect witness is made known to the access points. This relates to the the property of backwards unlinkability that was introduced in [31, 1, 27]. We note that in these papers backwards unlinkability is only partially achieved as time is divided into epochs and within an epoch all user transcripts would be traceable. In contrast in our setting of anonymous identification with suspect tracking we explicitly require only live checking of suspects, i.e., an access point can check exactly one suspicion per protocol invocation and past protocol transcripts reveal no tracking information.

2 Preliminaries

Notation. We write $s \overset{\mathtt{r}}{\leftarrow} S$ to denote randomly selecting s from a finite set S, and $s \overset{\mathtt{d}}{\leftarrow} S$ to denote selecting s from a finite set S according to distribution $D(S)$.

Smooth Projective Hashing for Hard Partitioned Subset Membership Problems. Smooth projective hashing was introduced by Cramer and Shoup for constructing encryption schemes [13]. Later it was used to obtain efficient password-based authenticated key-exchange protocols [21] and further to achieve stronger security in the UC framework [10]. It is also used to build efficient oblivious transfer protocols [24]. Loosely speaking, a projective hashing family is a family of hash functions that can be computed in two ways: either use the (secret) hashing key to compute the function on every point in its domain, or use the (public) projected key to compute the function only on a specified subset of the domain. Further we say such a family is "smooth" if the value of the function on any point outside the specified subset is independent of the projected key. Here we use a variant adopted by Gennaro and Lindell [21].

In our setting, a hash family is defined as a tuple $\mathcal{H} = (\mathsf{paragen}, \mathsf{sampler}, G, H, K, \alpha, S, J)$ that satisfies the following:

- The parameter generation algorithm $\mathsf{paragen}(\cdot)$ takes as input a security parameter and returns the parameter $\Lambda \leftarrow \mathsf{paragen}(1^\lambda)$, where $\Lambda = (X, L, W, R)$, and X, L, W are finite non-empty sets such that $L \subset X$. $R \subseteq X \times W$ is a binary relation, where for all $x \in X$ there exists $w \in W$ such that $(x, w) \in R$ iff $x \in L$.
- Further we assume that the set X can be written as a Cartesian product $C \times I$ and it is partitioned into disjoint subsets $X(i)$ as follows: each element $x \in X$ is in the form (c, i), where $i \in I$ will be called the index and $c \in C$ will be called the content. We define $X(i) \overset{\mathsf{def}}{=} C \times \{i\}$, i.e., the subset of pairs in X of the form (c, i) with i is fixed. Accordingly, we define $L(i)$ the subset of pairs in the language L of the form (c, i).
- Given Λ as a parameter, the random variable $\mathsf{sampler}(\Lambda, i)$ is a triple (x, x', w) such that $x \in L(i), x' \in X(i) \setminus L(i)$, $(x, w) \in R$. We will also use the notation $(x, w) \overset{\mathsf{d}}{\leftarrow} L(i)$ and $x' \overset{\mathsf{d}}{\leftarrow} X(i) \setminus L(i)$ respectively to denote the operation of the $\mathsf{sampler}(\cdot)$ procedure.
- G is a finite non-empty set, $H = \{H_k\}_{k \in K}$ is a collection of functions indexed by K where $H_k : X \to G$; S be a finite non-empty set, $\alpha : K \times C \to S$ and $J = \{J_s\}_{s \in S}$ is a collection of functions where $J_s : X \times W \to G$.

Definition 1. *The hash family* $\mathcal{H} = (\mathsf{paragen}, \mathsf{sampler}, G, H, K, \alpha, S, J)$ *is a* hard smooth projective hash family *if the following properties hold:*

Projection: *For any* $((c, i), w) \in R$
$\Pr[\Lambda \leftarrow \mathsf{paragen}(1^\lambda); k \overset{\mathtt{r}}{\leftarrow} K; s \leftarrow \alpha(k, c) : H_k(c, i) = J_s(c, i, w)] \geq 1 - \mathsf{negl}(\lambda)$

Smoothness: *For any* $(c, i) \in X \setminus L$*, the following two distributions are statistically indistinguishable:*
$\{\Lambda \leftarrow \mathsf{paragen}(1^\lambda); k \overset{\mathtt{r}}{\leftarrow} K; s \leftarrow \alpha(k, c); g \leftarrow H_k(x) : (\Lambda, c, i, s, g)\}$
$\{\Lambda \leftarrow \mathsf{paragen}(1^\lambda); k \overset{\mathtt{r}}{\leftarrow} K; s \leftarrow \alpha(k, c); g \overset{\mathtt{r}}{\leftarrow} G : (\Lambda, c, i, s, g)\}$

Hard partitioned subset membership: *For every* $i \in I$*, the following two distributions are computationally indistinguishable:*
$\{\Lambda \leftarrow \mathsf{paragen}(1^\lambda); ((c, i), w) \overset{\mathsf{d}}{\leftarrow} L(i) : (\Lambda, c, i)\}$
$\{\Lambda \leftarrow \mathsf{paragen}(1^\lambda); (c, i) \overset{\mathsf{d}}{\leftarrow} X(i) \setminus L(i) : (\Lambda, c, i)\}$

Gennaro and Lindell show that the properties above have the following implication:

Lemma 1 (Corollary 3.6 in [21]). *Given a hard smooth projective hash family* $\mathcal{H} = (\mathsf{paragen}, \mathsf{sampler},\ G, H, K, \alpha, S, J)$*, for any* $i \in I$*, the* *following two distributions are computationally indistinguishable:*

$$\{\Lambda \leftarrow \mathsf{paragen}(1^\lambda); ((c,i),w) \overset{\mathsf{d}}{\leftarrow} L(i); k \overset{\mathsf{r}}{\leftarrow} K; s \leftarrow \alpha(k,c); g \leftarrow H_k(c,i) : (\Lambda, c, i, s, g)\}$$

$$\{\Lambda \leftarrow \mathsf{paragen}(1^\lambda); ((c,i),w) \overset{\mathsf{d}}{\leftarrow} L(i); k \overset{\mathsf{r}}{\leftarrow} K; s \leftarrow \alpha(k,c); g \overset{\mathsf{r}}{\leftarrow} G : (\Lambda, c, i, s, g)\}$$

Hard Smooth Projective Hashing in the context of Public-Key Encryption. In our protocol design, we consider hard smooth projective hashing in the context of encryption schemes. Given a CPA-secure public key encryption scheme $\mathcal{E} = (\mathsf{gen}, \mathsf{enc}, \mathsf{dec})$, we define $\mathsf{paragen}$ and $\mathsf{sampler}$ algorithms for a hash family $\mathcal{H}$ as follows:

- The parameter generation algorithm $\mathsf{paragen}()$ operates as follows: first run a key pair $(pk, sk) \leftarrow \mathsf{gen}(1^\lambda)$; a ciphertext for a plaintext $m \in M$ can be computed by $c = \mathsf{enc}(pk; m; r)$ where $r \overset{\mathsf{r}}{\leftarrow} U$, and M is the plaintext space and U the random coins space; Let C_{pk} be the space of all ciphertexts based on public key pk; define $X \overset{\mathsf{def}}{=} C_{pk} \times M$, and $L \overset{\mathsf{def}}{=} \{(c,m) \in X \mid \exists r \text{ s.t. } c = \mathsf{enc}(pk; m; r)\}$. Note that the witness space $W = U$.
  Furthermore, define the partitioning of X to be by index $m \in M$, i.e., $X(m) \overset{\mathsf{def}}{=} C_{pk} \times \{m\}$, and $L(m) \overset{\mathsf{def}}{=} \{(c,m) \in X(m) \mid \exists r \text{ s.t. } c = \mathsf{enc}(pk; m; r)\}$.
- The sample procedure $\mathsf{sampler}()$ operates as follows: sample $((c,m), r) \in L(m)$ by computing $c \leftarrow \mathsf{enc}(pk; m; r)$ where $r \overset{\mathsf{r}}{\leftarrow} U$ is the witness; sample $(c,m) \in X(m) \setminus L(m)$ by computing $c \leftarrow \mathsf{enc}(pk; \tilde{m}; r)$ where $r \overset{\mathsf{r}}{\leftarrow} U$, $\tilde{m} \in M$, and $\tilde{m} \neq m$.

If we further can define (H, K, G, α, S, J) where $H_k : C_{pk} \times M \to G$, $J_s : C_{pk} \times M \times U \to G$, and $\alpha : K \times C_{pk} \to S$, to satisfy the properties of projection and smoothness, then we have a hard smooth projective hashing $\mathcal{H} = (\mathsf{paragen}, \mathsf{sampler}, G, H, K, \alpha, S, J)$. In [21], several concrete hard smooth projective hashing in the context of ElGamal encryption, Cramer-Shoup encryptions, and others are constructed. In section 3.3, we give a concrete construction in the context of Linear ElGamal encryption.

3 Zero-Knowledge with Witness Elimination

3.1 Functionality $\mathcal{F}_{\mathbf{ZKWE}}$

Here we introduce our new ZK functionality, *zero-knowledge with witness elimination*, $\mathcal{F}_{\text{ZKWE}}^{R,Q}$, in figure 1, where R is the ZK relation and Q is the elimination relation. The functionality interacts with an ideal adversary $\mathcal{S}$ and two parties, the prover P and the verifier T; the identities of both parties are encoded in session identifier *sid*. Intuitively $\mathcal{F}_{\text{ZKWE}}^{R,Q}$ can be viewed as an augmented ZK functionality with a "$\neg Q$-test" based on the private input of the verifier. Compared to the standard $\mathcal{F}_{\text{ZK}}$ functionality, here the functionality will receive from prover P an input (x, w), and will receive from verifier T a secret input w'. Once both inputs are given, T is supposed to eventually receive a bit φ, where $\varphi = 1$ if and only if both $(x, w) \in R$ and $(w, w') \notin Q$ hold. The functionality blocks the secret inputs w and w' as well as the outcome of the test $(w, w') \in Q$ from the ideal world adversary $\mathcal{S}$; it leaks though a bit ϕ to $\mathcal{S}$ where $\phi = 1$ if and only if $(x, w) \in R$ (cf. the standard ZK functionality, [7, 11]).

Functionality $\mathcal{F}_{\text{ZKWE}}^{R,Q}$

$\mathcal{F}_{\text{ZKWE}}^{R,Q}$ is parameterized with the ZK relation R and the elimination relation Q.

– Upon receiving $(\textsc{Prove}, sid, \langle x, w\rangle)$ from party P where $sid = (P, T, sid')$, record $\langle P, x, w\rangle$, send $(\textsc{LeakProve}, sid, \langle x, \phi\rangle, P)$ to the adversary, where $\phi = 1$ if $(x, w) \in R$, and $\phi = 0$ otherwise. Ignore future $(\textsc{Prove}, \ldots)$ inputs.
– Upon receiving $(\textsc{Verify}, sid, w')$ from party T where $sid = (P, T, sid')$, record $\langle T, w'\rangle$, send $(\textsc{LeakVerify}, sid, T)$ to the adversary. Ignore future $(\textsc{Verify}, \ldots)$ inputs.
– Upon receiving $(\textsc{InflVerify}, sid)$ from the adversary, if $(x, w) \in R$ and $(w, w') \notin Q$, then send $(\textsc{VerifyReturn}, sid, 1)$ to party T. Else if $(x, w) \notin R$ or $(w, w') \in Q$, then send $(\textsc{VerifyReturn}, sid, 0)$ to party T.

Fig. 1. Ideal functionality of zero-knowledge for the relation R with witness elimination based on the relation Q.

Some comments are in place:

– While one may use arbitrary relations Q for witness elimination, here we will be focusing on relations Q that capture some sort of equality test, either of the whole witness or a substring of the witness. This is consistent with our motivation in applying witness elimination to solve the suspect tracking problem in identification schemes. Nevertheless, more general elimination relations may be defined and could be potentially useful in other settings.
– Under the assumption that there is a way for any w to sample w' such that the event $(w, w') \in Q$ happens with negligible probability,

one can simulate the $\mathcal{F}_{\mathrm{ZK}}$ ideal functionality using an ideal $\mathcal{F}_{\mathrm{ZKWE}}$ functionality box. To see this observe that the verifier can play the role of T and select T's input w' at random. The assumption on Q needed for the simulation can be seen to be easily satisfied by the substring equality relations Q we consider here.

- Our formalization of $\mathcal{F}_{\mathrm{ZKWE}}$ ensures that as long as no party is corrupted the protocol transcript by itself should not leak any information about the relation of w, w'. This as an essential property as witness elimination should not be applied to protocols transcripts that are not "live." Moreover, even if the party T is corrupted the protocol transcript should enable a corrupted T to test the predicate $Q(w, \cdot)$ for **no more** than a single candidate witness. This property is similar to the property of resistance against off-line dictionary attacks in password-based key exchange schemes where an eavesdropper should not be able to use a protocol transcript to scan for the correct password.
- In our formalization of $\mathcal{F}_{\mathrm{ZKWE}}$ the adversary $\mathcal{S}$ learns whether the prover uses a valid witness or not. Intuitively this suggests that we allow the protocols that realize $\mathcal{F}_{\mathrm{ZKWE}}$ to have a method for a (dishonest) party T to extract this fact from a protocol interaction with an honest prover. This is consistent with the fact that we will only consider elimination relations Q for which the $\mathcal{F}^{R,Q}_{\mathrm{ZKWE}}$ functionality simulates the $\mathcal{F}^{R}_{\mathrm{ZK}}$ ideal functionality and thus this does not affect the intended security properties of the protocol (the prover P is aware that the verifier T can easily learn whether she possesses a valid witness or not, cf. the second bullet above).

3.2 Generic construction based on smooth projective hashing

In this section we present a construction for $\mathcal{F}^{R,Q}_{\mathrm{ZKWE}}$ that applies to the setting when the elimination relation Q is either the equality relation or it contains all elements of the form $\langle (w_1, w_2), (w'_1, w'_2) \rangle$ such that $w_2 = w'_2$ (i.e., Q is a "substring equality" relation). Based on this, in the remaining of the section we assume that the NP relation R contains pairs of the form $(x, (w_1, w_2))$ and the elimination relation Q tests the "w_2-part" of the witness (while it should be noted that the "w_1-part" could be empty).

Our construction is based on a CPA-secure encryption scheme $\mathcal{E} = (\mathsf{gen}, \mathsf{enc}, \mathsf{dec})$, a zero-knowledge proof of membership (ZKPM) scheme $\mathcal{P} = (\mathsf{setup}, \mathsf{P}, \mathsf{V}, \mathsf{S})$, and a hard smooth projective hashing $\mathcal{H} = (\mathsf{paragen}, \mathsf{sampler}, G, H, K, \alpha, S, J)$ in the context of a CPA-secure encryption scheme $\mathcal{E}' = (\mathsf{gen}', \mathsf{enc}', \mathsf{dec}')$. Recall that both P and T are supposed to keep their inputs private. P's input is the witness to the ZK relation that she is supposed to demonstrate knowledge of while T's input determines the elimination relation. In accordance to the above our construction includes: *(1)* an encryption of the prover's input under $\mathcal{E}$ and the verifier's input under $\mathcal{E}'$, *(2)* an inequality test subprotocol using the smooth projective hash $\mathcal{H}$, and *(3)* a ZKPM for consistency proof that the above *(1)* and *(2)* components are consistent and at the same time the input of the prover satisfies the ZK-relation R. As mentioned, to implement the (one-sided) inequality test, we use the hard smooth projective hashing in the context of the public-key encryption scheme $\mathcal{E}'$. If the party P's input witness is indeed matched by the verifier T, i.e., $w_2 = w_2'$, then the projection property of the hashing will guarantee that $\kappa = \kappa'$ holds (where κ, κ' are the respective hash values with κ being transmitted by the prover to the verifier), and otherwise if $w_2 \neq w_2'$ then $\kappa \neq \kappa'$. A corrupted P cannot learn T's input given the underlying public-key encryption $\mathcal{E}'$ is CPA-secure. A corrupted T on the other hand cannot learn P's witness in the case that $w_2 \neq w_2'$ given the smoothness property of the hashing as well as the CPA property of $\mathcal{E}$. Of course T will learn (part of) P's witness in the case $w_2 = w_2'$; nevertheless, any polynomial-time bounded eavesdropper will not be able to extract any information about the exchanged witnesses (cf. lemma 1). In figure 2 we present the protocol in details.

Regarding the security of the construction we have the following:

Theorem 1. *Given that $\mathcal{E}$ is a CPA-secure encryption scheme, $\mathcal{H}$ is a family of hard smooth projective hash functions in the context of a CPA-secure encryption scheme $\mathcal{E}'$, and $\mathcal{P}$ is a zero-knowledge proof of membership, then the construction in figure 2 realizes functionality $\mathcal{F}^{R,Q}_{\mathrm{ZKWE}}$ against non-adaptive adversaries in the $\mathcal{F}_{\mathrm{CRS}}$-hybrid world.*

In order to prove the theorem, we describe first an ideal world simulator $\mathcal{S}$ as below; then we show for this simulator that for any PPT environment $\mathcal{Z}$, $\mathrm{EXEC}^{\mathcal{F}_{\mathrm{CRS}}}_{\pi,\mathcal{A}_d,\mathcal{Z}} \approx \mathrm{EXEC}^{\mathcal{F}^{R,Q}_{\mathrm{ZKWE}}}_{\pi_d,\mathcal{S},\mathcal{Z}}$ (the proof can be found in the full version).

The construction of simulator. The simulator $\mathcal{S}$ first runs the key generation algorithms of both the encryption schemes $\mathcal{E}'$, $\mathcal{E}$ and the proof system $\mathcal{P}$ to obtain key pairs, i.e., $(pk', sk') \leftarrow \mathsf{gen}'(1^\lambda)$, $(pk, sk) \leftarrow \mathsf{gen}(1^\lambda)$, and $(\rho, \tau) \leftarrow \mathsf{setup}(1^\lambda)$; then $\mathcal{S}$ sets $crs \leftarrow (pk', pk, \rho)$ and the corresponding trapdoor $td \leftarrow (sk', sk, \tau)$. Then $\mathcal{S}$ initializes $\mathcal{A}_d$ by giving it crs as the common reference string. Now the simulator $\mathcal{S}$ interacts with the environment $\mathcal{Z}$, the functionality $\mathcal{F}^{R,Q}_{\mathrm{ZKWE}}$ and its subroutine $\mathcal{A}_d$.

Protocol π for zero-knowledge with witness elimination for relations R and Q

Common reference string: $crs = (pk', pk, \rho)$, where pk' and pk are public keys of encryption schemes $\mathcal{E}'$ and $\mathcal{E}$ respectively, and ρ is a reference string of ZKPM scheme $\mathcal{P}$.

Protocol steps:

Upon receiving $(\textsc{Verify}, sid, w_2')$ from the environment, where $sid = (P, T, sid')$, party T selects $r' \overset{r}{\leftarrow} U'$ and computes $c' \leftarrow \mathsf{enc}'(pk'; w_2'; r')$, and sends $(\mathtt{move1}, c')$ to party P.

Upon receiving $(\textsc{Prove}, sid, \langle x, w_1, w_2 \rangle)$ from the environment, party P first checks if $(x, (w_1, w_2)) \in^? R$ and waits for a `move1` message from party T; after receiving `move1` message,

- if $(x, (w_1, w_2)) \notin R$ then party P sends party T a message $(\mathtt{move2}, \text{“no valid proof”})$;
- else if $(x, (w_1, w_2)) \in R$ then party P sends party T a message $(\mathtt{move2}, s, \kappa)$, where $\kappa \leftarrow H_k(c', w_2)$, $s \leftarrow \alpha(k, c')$, $k \overset{r}{\leftarrow} K$; then party P computes $c \leftarrow \mathsf{enc}(pk; w_1, w_2; r)$ where $r \overset{r}{\leftarrow} U$; now parties P and T play the roles of prover and verifier respectively to run a ZKPM subprotocol

$$\mathsf{P}(w_1, w_2, k) \xleftrightarrow{(x, c', c, \kappa) \in L_{R'}} \mathsf{V}()$$

i.e, party P proves to party T that based on crs, the statement $(x, c', c, \kappa) \in L_{R'}$ where $L_{R'} = \{(x, c', c, \kappa) | \exists (w_1, w_2, k, r) \text{ s.t. } (x, (w_1, w_2)) \in R \wedge \kappa = H_k(c', w_2) \wedge\ c = \mathsf{enc}(pk; w_1, w_2; r)\}$.

Upon receiving $(\mathtt{move2}, \text{“no valid proof”})$ from party P, party T returns $(\textsc{VerifyReturn}, sid, 0)$ to the environment. Else if receiving $(\mathtt{move2}, s, \kappa)$, party T computes $\kappa' \leftarrow J_s(c', w_2'; r')$; and if $\kappa \neq \kappa'$ and also party T accepts the proof in the subprotocol above, then party T returns $(\textsc{VerifyReturn}, sid, 1)$ to the environment; otherwise returns $(\textsc{VerifyReturn}, sid, 0)$ to the environment.

Fig. 2. Generic construction of zero-knowledge with witness elimination based on a CPA-secure encryption scheme $\mathcal{E} = (\mathsf{gen}, \mathsf{enc}, \mathsf{dec})$, and a zero-knowledge proof of membership scheme $\mathcal{P} = (\mathsf{setup}, \mathsf{P}, \mathsf{V}, \mathsf{S})$, as well as a hard smooth projective hashing $\mathcal{H} = (\mathsf{paragen}, \mathsf{sampler}, G, H, K, \alpha, S, J)$ in the context of a CPA-secure encryption scheme $\mathcal{E}' = (\mathsf{gen}', \mathsf{enc}', \mathsf{dec}')$.

The simulator $\mathcal{S}$ simulates the protocol transcripts by following the protocol π on behalf of the honest parties. Since the secret inputs of the honest parties are not known, the simulator uses randomly selected $\tilde{w}_2'$ and $\tilde{w}$ as the witnesses to compute c' and κ, c, and further, the simulator runs S with V to simulate the ZKPM subprotocol transcripts. We give details below. Note that we only consider non-adaptive corruptions.

After receiving $(\textsc{LeakProve}, \mathit{sid}, \langle x, 1\rangle, P)$ from the functionality (party P is honest), and the `move1` message from party T including c', the simulator $\mathcal{S}$ uses a randomly selected $\tilde{w} = (\tilde{w}_1, \tilde{w}_2)$ (since the real witness $w = (w_1, w_2)$ is blocked by the functionality) to compute κ and c, and runs S with V (instead of P with V) to simulate the ZKPM subprotocol proof transcripts. If on the other hand it receives $(\textsc{LeakProve}, \mathit{sid}, \langle x, 0\rangle, P)$ from the functionality and the `move1` message from T, $\mathcal{S}$ responds T with a `move2` message "no valid proof".

After receiving $(\textsc{LeakVerify}, \mathit{sid}, T)$ from the functionality (party T is honest), the simulator $\mathcal{S}$ uses a randomly selected $\tilde{w}_2'$ (since the real witness w_2' is blocked by the functionality) to compute c', and sends c' as the `move1` message to party P. Later when it receives the `move2` message from party P, $\mathcal{S}$ sends $(\textsc{InflVerify}, \mathit{sid})$ to the functionality.

We next consider the case that party T is corrupted. When $\mathcal{S}$ receives $(\textsc{LeakProve}, \mathit{sid}, \langle x, 1\rangle, P)$ from the functionality and a `move1` message including c' from the corrupted party T, the simulator uses sk' to decrypt c' into w_2', and then sends the input $(\textsc{Verify}, \mathit{sid}, w_2')$ in the name of T to the functionality; after receiving $(\textsc{LeakVerify}, \mathit{sid}, T)$ from the functionality, $\mathcal{S}$ sends $(\textsc{InflVerify}, \mathit{sid})$ to the functionality; if the functionality returns $(\textsc{VerifyReturn}, \mathit{sid}, 1)$ which means $w_2' \neq w_2$ where w_2 is from the actual witness of the prover, the simulator randomly selects $\breve{w} = (\breve{w}_1, \breve{w}_2)$ to finish the simulation of the P protocol; if the functionality returns $(\textsc{VerifyReturn}, \mathit{sid}, 0)$ which means $w_2' = w_2$ where w_2 is from the real witness, the simulator randomly selects $\breve{w}_1$, and uses $\breve{w} = (\breve{w}_1, w_2)$ to finish the simulation. On the other hand, if it receives $(\textsc{LeakProve}, \mathit{sid}, \langle x, 0\rangle, P)$ from the functionality and a `move1` message, $\mathcal{S}$ simulates the P by responding with "no valid proof" as the `move2` message.

We now consider the case that party P is corrupted. If $\mathcal{S}$ receives the message $(\texttt{move2}, s, \kappa)$ from the corrupted party P, and after executing the ZKPM subprotocol the party V returns 1 which means party T accepts the subprotocol proof that $(x, c', c, \kappa) \in L_{R'}$, the simulator uses sk to decrypt c into w and sends input $(\textsc{Prove}, \mathit{sid}, \langle x, w\rangle)$ in the name of P to the functionality; else if V returns 0 which means party T rejects the subprotocol proof, then the simulator sends input $(\textsc{Prove}, sid, \langle x, \perp\rangle)$ where $(x, \perp) \notin R$. Further, if $\mathcal{S}$ receives $(\texttt{move2}, \text{“no valid proof”})$ from the corrupted P, the simulator also sends input $(\textsc{Prove}, sid, \langle x, \perp\rangle)$ to functionality. After it receives $(\textsc{LeakProve}, sid, \langle x, \phi\rangle)$ from the functionality, $\mathcal{S}$ sends $(\textsc{InflVerify}, sid)$ to the functionality, and the functionality returns $(\textsc{VerifyReturn}, sid, \varphi)$ to the dummy T according to its program. This completes the description of the simulator $\mathcal{S}$.

3.3 Illustrative efficient construction

In this subsection, we instantiate the generic construction to obtain an explicit practical protocol for a zero-knowledge proof with witness elimination. The zero-knowledge proof we develop is for proving possession of a non-adaptive Boneh-Boyen digital signature [3]. We first introduce a building block, a hard smooth projective hashing in the context of linear ElGamal encryption [4], and then give the protocol.

Our construction yields immediately an anonymous identification scheme with suspect tracking. This can be seen as follows: each user in the system will hold a BB signature on a random message while the public-key of the BB signature will be publicly available. To authenticate itself to a verifier, a prover will engage in a proof of knowledge of a signature. Note that based on the non-adaptive unforgeability of BB signatures it is impossible for any coalition of users to obtain an additional signature. Observe now that if the system manager wishes to track a suspect it may disclose the message-signature pair that was assigned to the particular user. This will enable any verifier to employ witness elimination and thus check that any prover that engages in the interaction does not possess the suspect message-signature pair.

Bilinear Groups. Let $\mathbb{G} = \langle g\rangle$ be a cyclic group of prime order p such that $\check{\mathsf{e}} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_t$ is a bilinear map, i.e., for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, it holds that $\check{\mathsf{e}}(u^a, v^b) = \check{\mathsf{e}}(u, v)^{ab}$ and $\check{\mathsf{e}}$ is non-trivial, i.e., $\check{\mathsf{e}}(g, g) \neq 1$. Note that $|\mathbb{G}_t| = p$.

Linear Encryption. Boneh et al. [4] proposed a variant of ElGamal encryption, called, Linear Encryption that is suitable for groups over which the DDH assumption fails.
Key Generation: $(pk, sk) \leftarrow \mathsf{gen}(1^\lambda)$, where $pk = (u, v, w)$ and $sk = (y, z)$, and $u, v, w \in \mathbb{G}$ and $y, z \in \mathbb{Z}_p$ such that $u^y = v^z = w$.
Encryption: $c \leftarrow \mathsf{enc}(pk; m; r)$, where $m \in \mathbb{G}$, $r = (\eta, \theta)$ with $\eta, \theta \xleftarrow{\mathtt{r}} \mathbb{Z}_p$, and $c = (u^\eta, v^\theta, w^{\eta+\theta} m)$.

Decryption: $m \leftarrow \mathsf{dec}(pk, sk; c)$, where $c = (U, V, W)$, $sk = (y, z)$, and $m = \frac{W}{U^y \cdot V^z}$.

The Linear encryption is CPA-secure under the Decision Linear Diffie-Hellman assumption [4].

Definition 2 (Decision Linear Diffie-Hellman Assumption). *We say that the Decision Linear Diffie-Hellman assumption holds in $\mathbb{G}$ if for all PPT distinguisher $\mathcal{A}$ we have*

$$\left| \begin{array}{l} \Pr[u, v, w \xleftarrow{r} \mathbb{G}; \beta, \gamma \xleftarrow{r} \mathbb{Z}_p : \mathcal{A}(u, v, w, u^\beta, v^\gamma, w^{\beta+\gamma}) = 1] \\ \quad - \Pr[u, v, w, \chi \xleftarrow{r} \mathbb{G}; \beta, \gamma \xleftarrow{r} \mathbb{Z}_p : \mathcal{A}(u, v, w, u^\beta, v^\gamma, \chi) = 1] \end{array} \right| \leq \mathsf{negl}(\lambda)$$

Hard smooth projective hashing in the context of linear ElGamal encryption. We now define a smooth projective hash function LEH for this encryption scheme. Here $C_{pk} \overset{\mathsf{def}}{=} (\mathbb{G})^3$ and $M \overset{\mathsf{def}}{=} \mathbb{G}$, so $X = (\mathbb{G})^4$; and $W = (\mathbb{Z}_p)^2$ and $G = \mathbb{G}$; the key space is defined by $K = (\mathbb{Z}_p)^3$, i.e., a key is a triple $k = (\beta, \gamma, \delta)$, where $\beta, \gamma, \delta \xleftarrow{r} \mathbb{Z}_p$; the key projection function α on input the key $k = (\beta, \gamma, \delta)$ and ciphertext $c = (U, V, W)$ is defined by $s = (s_1, s_2) = (u^\beta w^\delta, v^\gamma w^\delta)$; function $H_k : (\mathbb{G})^4 \to \mathbb{G}$ is defined as $H_k((U, V, W), m) = U^\beta V^\gamma (W/m)^\delta$; function $J_s : (\mathbb{G})^4 \times (\mathbb{Z}_p)^2 \to \mathbb{G}$ is defined as $J_s((U, V, W), m, (\eta, \theta)) = (s_1)^\eta (s_2)^\theta$.

Lemma 2. *LEH is a hard smooth projective hash function.*

The zero-knowledge with witness elimination protocol. In our protocol, party P keeps a witness including two parts m and σ such that the relation $\breve{\mathsf{e}}(g^m X, \sigma) = \breve{\mathsf{e}}(g, g)$ which can be viewed as a non-adaptive BB message-signature pair with public parameter $(p, g, X, \mathbb{G}, \mathbb{G}_t, \breve{\mathsf{e}})$ and signing secret x, where $X = g^x$. The relation Q for which the witness elimination is performed is an equality test on the signature part of the message-signature pair. Note that in the non-adaptive BB-signature there is a correspondence between messages and signatures and thus Q amounts to simply an equality relation.

The inequality test subprotocol is based on a hard smooth projective hashing in the context of linear ElGamal encryption introduced in the previous subsection. Then we use two encryption schemes to encrypt P's witness, i.e., a Paillier [29] encryption to encrypt m into E, and a linear ElGamal encryption to encrypt σ into $\langle U, V, W \rangle$. We construct a three-move Sigma-protocol to prove: (1) m and σ is a valid non-adaptive BB message-signature pair; (2) the inequality subprotocol is consistent with the statement in (1); and (3) E and $\langle U, V, W\rangle$ are cipertexts for m based on Paillier encryption and for σ based on linear ElGamal encryption respectively. Further to defend against a dishonest verifier, we use a technique of [14] to wrap up the above Sigma-protocol by using a commitment scheme. The protocol is presented in full details in figure 3.

Acknowledgement: We thank Juan Garay for helpful discussions, and the current title was suggested by him. We also thank the anonymous reviewers for valuable comments.

References

1. G. Ateniese, D. X. Song, and G. Tsudik. Quasi-efficient revocation in group signatures. In M. Blaze, editor, *Financial Cryptography*, volume 2357 of *LNCS*, pages 183–197. Springer, 2002.
2. M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *STOC*, pages 103–112. ACM, 1988.
3. D. Boneh and X. Boyen. Short signatures without random oracles. In C. Cachin and J. Camenisch, editors, *EUROCRYPT*, volume 3027 of *LNCS*, pages 56–73. Springer, 2004.
4. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. K. Franklin, editor, *CRYPTO*, volume 3152 of *LNCS*, pages 41–55. Springer, 2004.
5. D. Boneh and H. Shacham. Group signatures with verifier-local revocation. In V. Atluri, B. Pfitzmann, and P. D. McDaniel, editors, *ACM Conference on Computer and Communications Security*, pages 168–177. ACM, 2004.
6. J. Camenisch. Group signature schemes and payment systems based on the discrete logarithm problem. *ETH Series in Information Security an Cryptography*, 2, 1998. PhD thesis available at `http://www.zurich.ibm.com/~jca/papers/diss.ps`.
7. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145. IEEE Computer Society, 2001.
8. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Cryptology ePrint Archive, Report 2000/067*, December 2005. Latest version at `http://eprint.iacr.org/2000/067/`.
9. R. Canetti, O. Goldreich, S. Goldwasser, and S. Micali. Resettable zero-knowledge (extended abstract). In *STOC*, pages 235–244, 2000. Full version at `http://eprint.iacr.org/1999/022`.
10. R. Canetti, S. Halevi, J. Katz, Y. Lindell, and P. D. MacKenzie. Universally composable password-based key exchange. In R. Cramer, editor, *EUROCRYPT*, volume 3494 of *LNCS*, pages 404–421. Springer, 2005. Full version at `http://eprint.iacr.org/2005/196`.
11. R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *STOC*, pages 494–503. ACM, 2002. Full version at `http://eprint.iacr.org/2002/140`.
12. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Y. Desmedt, editor, *CRYPTO*, volume 839 of *LNCS*, pages 174–187. Springer, 1994.

$$crs = (q, \mathbf{g}, \mathbf{h}, \mathbf{G}, \mathsf{hash}; \mathsf{n}, \mathsf{g}; p, g, u, v, w, X, \mathbb{G}, \mathbb{G}_t, \check{\mathsf{e}})$$

$\boxed{P}$ $\boxed{T}$

P: $\texttt{input} = (m, \sigma)$ s.t. $\check{\mathsf{e}}(g^m X, \sigma) = \check{\mathsf{e}}(g, g)$

T: $\texttt{input} = \sigma'$

T:
$\eta', \theta' \xleftarrow{\mathtt{r}} \mathbb{Z}_p;$
$U' \leftarrow u^{\eta'};\ V' \leftarrow v^{\theta'};\ W' \leftarrow w^{\eta'+\theta'}\sigma';$

$$\xleftarrow{U', V', W'}$$

P:
$\beta, \gamma, \delta, \zeta_\beta, \zeta_\gamma, \zeta_\delta \xleftarrow{\mathtt{r}} \mathbb{Z}_p;$
$s_1 \leftarrow u^\beta w^\delta;\ s_2 \leftarrow v^\gamma w^\delta;$
$\kappa \leftarrow (U')^\beta (V')^\gamma (\frac{W'}{\sigma})^\delta;$

$$\xrightarrow{s_1, s_2, \kappa}$$

T:
$\kappa' \leftarrow (s_1)^{\eta'} (s_2)^{\theta'};$
$\kappa =^? \kappa';$

P:
$\zeta_m \xleftarrow{\mathtt{r}} \pm[0, 2^{\ell_0+\ell_d+\ell_p}];\ \chi, \zeta_\chi \xleftarrow{\mathtt{r}} \mathbb{Z}_{\mathsf{n}}^*;$
$E \leftarrow \mathsf{g}^m \chi^{\mathsf{n}} \bmod \mathsf{n}^2;$
$B_1 \leftarrow \mathsf{g}^{\zeta_m} (\zeta_\chi)^{\mathsf{n}} \bmod \mathsf{n}^2;$
$\eta, \theta, \zeta_\eta, \zeta_\theta, \zeta_{m\eta}, \zeta_{m\theta}, \zeta_{\delta\eta}, \zeta_{\delta\theta} \xleftarrow{\mathtt{r}} \mathbb{Z}_p;$
$U \leftarrow u^\eta;\ V \leftarrow v^\theta;\ W \leftarrow w^{\eta+\theta}\sigma;$
$B_2 \leftarrow u^{\zeta_\eta};\ B_3 \leftarrow v^{\zeta_\theta};$
$B_4 \leftarrow (U')^{\zeta_\beta} (V')^{\zeta_\gamma} (W' W^{-1})^{\zeta_\delta} w^{\zeta_{\delta\eta} + \zeta_{\delta\theta}};$
$B_5 \leftarrow \check{\mathsf{e}}(g, W)^{\zeta_m} \check{\mathsf{e}}(g, w)^{-\zeta_{m\eta} - \zeta_{m\theta}} \check{\mathsf{e}}(X, w)^{-\zeta_\eta - \zeta_\theta};$
$\omega \leftarrow \mathsf{hash}(B_1, B_2, B_3, B_4, B_5);$
$\mu \xleftarrow{\mathtt{r}} \mathbb{Z}_q;\ C \leftarrow \mathbf{g}^\omega \mathbf{h}^\mu;$

$$\xrightarrow{E, U, V, W, C}$$

T:
$d \xleftarrow{\mathtt{r}} \{0, 1\}^{\ell_d};$

$$\xleftarrow{d}$$

P:
$\xi_\chi \leftarrow \zeta_\chi \cdot \chi^d \bmod \mathsf{n};\quad \xi_\eta \leftarrow \zeta_\eta + d \cdot \eta;$
$\xi_\theta \leftarrow \zeta_\theta + d \cdot \theta;\quad \xi_\delta \leftarrow \zeta_\delta + d \cdot \delta;$
$\xi_\beta \leftarrow \zeta_\beta + d \cdot \beta;\quad \xi_\gamma \leftarrow \zeta_\gamma + d \cdot \gamma;$
$\xi_{\delta\eta} \leftarrow \zeta_{\delta\eta} + d \cdot \delta\eta;$
$\xi_{\delta\theta} \leftarrow \zeta_{\delta\theta} + d \cdot \delta\theta;$
$\xi_m \leftarrow \zeta_m + d \cdot m$ (in $\mathbb{Z}$);
$\xi_{m\eta} \leftarrow \zeta_{m\eta} + d \cdot m\eta;$
$\xi_{m\theta} \leftarrow \zeta_{m\theta} + d \cdot m\theta;$
$\boldsymbol{\xi} \leftarrow (\xi_\chi, \xi_\eta, \xi_\theta, \xi_\delta, \xi_\beta, \xi_\gamma, \xi_{\delta\eta}, \xi_{\delta\theta}, \xi_m, \xi_{m\eta}, \xi_{m\theta});$
$\boldsymbol{B} \leftarrow (B_1, B_2, B_3, B_4, B_5);$

$$\xrightarrow{\boldsymbol{\xi}, \boldsymbol{B}, \mu}$$

T:
$\omega \leftarrow \mathsf{hash}(B_1, B_2, B_3, B_4, B_5); \mathbf{g}^\omega \mathbf{h}^\mu =^? C;$
$\xi_m \in^? \pm[0, 2^{\ell_0+\ell_d+\ell_p+1}]; E \in^? \mathbb{Z}_{\mathsf{n}^2}^*;$
$\mathsf{g}^{\xi_m} (\xi_\chi)^{\mathsf{n}} =^? B_1 E^d \bmod \mathsf{n}^2;$
$u^{\xi_\eta} =^? B_2 U^d;\ v^{\xi_\theta} =^? B_3 V^d;$
$(U')^{\xi_\beta} (V')^{\xi_\gamma} (W' W^{-1})^{\xi_\delta} w^{\xi_{\delta\eta} + \xi_{\delta\theta}} =^? B_4 \kappa^d;$
$\check{\mathsf{e}}(g, W)^{\xi_m} \check{\mathsf{e}}(g, w)^{-\xi_{m\eta} - \xi_{m\theta}} \check{\mathsf{e}}(X, w)^{-\xi_\eta - \xi_\theta}$
$=^? B_5 \check{\mathsf{e}}(g, g)^d \check{\mathsf{e}}(X, W)^{-d};$

Fig. 3. Concrete construction of zero-knowledge with witness elimination.

13. R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In L. R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *LNCS*, pages 45–64. Springer, 2002. Full version available at `http://www.shoup.net/papers/uhp.pdf`.
14. I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *EUROCRYPT*, volume 1807 of *LNCS*, pages 418–430. Springer, 2000.
15. I. Damgård, J. B. Nielsen, and D. Wichs. Isolated proofs of knowledge and isolated zero knowledge. In N. P. Smart, editor, *EUROCRYPT*, volume 4965 of *LNCS*, pages 509–526. Springer, 2008.
16. A. De Santis, G. Di Crescenzo, G. Persiano, and M. Yung. On monotone formula closure of szk. In *FOCS*, pages 454–465. IEEE, 1994.
17. D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000. Preliminary version appears at STOC 1991.
18. C. Dwork, M. Naor, and A. Sahai. Concurrent zero-knowledge. *J. ACM*, 51(6):851–898, 2004. Preliminary version appears at STOC 1998.
19. U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. In *STOC*, pages 416–426. ACM, 1990.
20. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *CRYPTO*, volume 263 of *LNCS*, pages 186–194. Springer, 1986.
21. R. Gennaro and Y. Lindell. A framework for password-based authenticated key exchange. *ACM Trans. Inf. Syst. Secur.*, 9(2):181–234, 2006. Preliminary version appears at Eurocrypt 2003.
22. O. Goldreich and H. Krawczyk. On the composition of zero-knowledge proof systems. *SIAM J. Comput.*, 25(1):169–192, 1996. Preliminary version appears at ICALP 1990.
23. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.
24. S. Halevi and Y. T. Kalai. Smooth projective hashing and two-message oblivious transfer. In *Cryptology ePrint Archive: Report 2007/118*, 2007. Preliminary version appears at Eurocrypt 2005.
25. J. Katz, R. Ostrovsky, and M. Yung. Efficient password-authenticated key exchange using human-memorable passwords. In B. Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *LNCS*, pages 475–494. Springer, 2001.
26. A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In C. Cachin and J. Camenisch, editors, *EUROCRYPT*, volume 3027 of *LNCS*, pages 571–589. Springer, 2004.
27. T. Nakanishi and N. Funabiki. Verifier-local revocation group signature schemes with backward unlinkability from bilinear maps. In B. K. Roy, editor, *ASIACRYPT*, volume 3788 of *LNCS*, pages 533–548. Springer, 2005.
28. M. Naor and B. Pinkas. Oblivious polynomial evaluation. *SIAM J. Comput.*, 35(5):1254–1281, 2006. Preliminary version appears at STOC 1999.
29. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *EUROCRYPT*, volume 1592 of *LNCS*, pages 223–238. Springer, 1999.
30. C.-P. Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.
31. D. X. Song. Practical forward secure group signature schemes. In *ACM Conference on Computer and Communications Security*, pages 225–234, 2001.