Adaptively Secure Threshold Cryptography: Introducing Concurrency, Removing Erasures

(Extended Abstract)

Stanisław Jarecki and Anna Lysyanskaya*

MIT Laboratory for Computer Science, Cambridge, MA 02139, USA
{stasio, anna}@theory.lcs.mit.edu

Abstract. This extended abstract is a concise presentation of two independent results by Lysyanskaya [Lys00] and Jarecki and Lysyanskaya [JL00]. We put forward two new measures of security for threshold schemes secure in the adaptive adversary model: security under concurrent composition [Lys00]; and security without the assumption of reliable erasure [JL00]. Using novel constructions and analytical tools, in both these settings, we exhibit efficient secure threshold protocols for a variety of cryptographic applications. In particular, based on the recent scheme by Cramer-Shoup, we construct adaptively secure threshold cryptosystems secure against adaptive chosen ciphertext attack under the DDH intractability assumption. Our techniques are also applicable to other cryptosystems and signature schemes, like RSA, DSS, and ElGamal.
Our techniques include the first efficient implementation, for a wide but special class of protocols, of secure channels in erasure-free adaptive model. Of independent interest, we introduce the notion of a *committed proof*.
Of independent interest, we present the notion of a *committed proof* [Lys00].

1 Introduction

Overview. The idea of threshold cryptography [Des87,DF89] is that a highly sensitive operation such as decryption or signing, can be performed by a group of cooperating servers in such a way that no minority of servers are able to perform the operation by themselves, nor are they be able to prevent the other servers from performing the operation when it is required. Thus, threshold protocols implement trusted entities, based on the assumption that only a fraction of a given set of dedicated servers can become corrupted. However, it is a challenging task to design protocols that are secure in the face of realistic attacks against the servers. In the present extended abstract we consider two such attacks for which no previous threshold schemes can be proven secure. The two attacks correspond to two quite limiting assumptions which were necessary for the previously known solutions, and which consequently hindered their applicability.

We consider a concurrent attack where the adversary tries to get an advantage by participating in several concurrent executions of the threshold protocol. Since previous schemes were not provably secure in this adversarial model, they were limited to sequential execution synchronized among all servers. We also consider an attack in which the entire history of a server's computation is recorded and

* Part of this research was carried out while the author was visiting IBM Zürich Research Laboratory.

becomes available to an adversary that corrupts this server. Since no schemes were provable in this model, they had to be executed on servers that could reliably erase their data. For both of these adversarial models, we devise novel techniques that allow us to implement efficient protocols that withstand them. We exemplify these techniques with threshold implementations of the Cramer-Shoup cryptosystem [CS98], which achieves the highest known level of security: security against adaptive chosen ciphertext attack. Furthermore, our techniques also yield efficient concurrent or erasure-free adaptively secure solutions to other schemes like RSA, DSS, and ElGamal.

History. For a long time, we knew only how to design threshold protocols secure in the so-called *static* adversary model where the adversary fixes the players that will be corrupted before the protocol starts. Recently, Canetti et al. [CGJ+99a] and Frankel et al. [FMY99a-b] exhibited the first threshold schemes secure and robust against the stronger and more realistic *adaptive* adversary, who chooses which players to corrupt at any time and based on any information he sees during the protocol. These results are important since it is known that the adaptive adversary is strictly stronger than the static one [CFGN96,Can98,CDD+99]. However, none of these adaptively secure protocols remained secure under concurrent composition, and they all required erasures. In addition, the cryptosystems and signature schemes implemented by these threshold schemes are not known to be provably secure under adaptive chosen ciphertext attack/adaptive chosen message attack. We remark that even though general multi-party computation results guarantee adaptive erasure-free distributed function evaluation [BGW88,CCD88,CDD+99,CFGN96], implementing threshold cryptography via these general techniques is impractical.

General model. We consider a network of n players and an adaptive adversary that can corrupt up to a minority $t < n/2$ of the players. The players have access to a reliable broadcast channel, there are insecure point-to-point links between each pair of them, and the message delivery is partially synchronous.

Concurrent model. We consider the *concurrent* setting, where many invocations of possibly the same threshold cryptosystem or signature scheme can be executed at the same time, and each of them must remain secure. This previously unexplored setting models an important property of threshold systems: the possibility of executing several protocols at the same time.

Erasure-free model. All of the threshold systems mentioned so far are implemented using erasures. That is to say, they are only secure if honest players can erase local data once it is no longer needed. However, secure erasure is hard to achieve in practice. On the hardware level, it is difficult to permanently erase information from hardware storage devices. On the system maintenance level, the need to erase data complicates standard computer system bookkeeping and backup procedures. Most serious problems arise on the operating systems level, since in order to securely erase the data, one needs to erase it from all the caches and from the part of the hard drive that was used for page swaps, etc. Di Crescenzo et al. [CFIJ99] discuss this problem and suggest a solution that enables erasures based on the assumption that some area of memory can indeed be securely erased. In contrast, we show that in the adaptively secure threshold setting it is possible to get rid of the need of secure data erasure altogether. We thus examine an *erasure-free* model, in which the adversary is effectively allowed to examine the entire history of the computation of a party it corrupts.

Techniques of independent interest. We introduce the notion of a *committed proof*, i.e. a zero-knowledge proof of an unknown statement [Lys00]. It was not known before that it was possible to prove a statement without revealing it to the verifier until the very last round of communication. Here we use such committed proofs to achieve threshold cryptosystems adaptively secure in the concurrent model. Another useful technique of independent interest that we put forward as an implementation of secure channels in the erasure-free model is our *receiver-non-committing encryption* scheme [JL00]. A non-committing encryption scheme has a property that there is a way to generate messages that look like ciphertexts but do not commit the players to any particular plaintext. We give a simple and efficient encryption scheme that is non-committing for the receiver under the decisional Diffie-Hellman intractability assumption.

Organization. In Section 2 we give an overview of our results and the most important techniques which allow us to achieve them. In Section 3 we present the notion and an implementation of a committed proof. Section 4 presents our non-committing encryption scheme. We then present our adaptive threshold protocols: Section 5 describes the basic building blocks of our solutions; sections 6 and 7 exemplify our techniques with two threshold implementations of the Cramer-Shoup cryptosystem: (1) an erasure-enabled protocol secure in concurrent composition; (2) an erasure-free protocol which is concurrently secure only under certain restrictions. For a more thorough treatment of our results pertaining to the concurrent model and to committed proofs, see the work of Lysyanskaya [Lys00]. For a more thorough treatment of our results pertaining to the erasure-free model and non-committing encryption, see Jarecki and Lysyanskaya [JL00].

2 Overview of our concurrent and erasure-free protocols

Definitions and goals. A threshold cryptosystem or signature scheme implemented by n players with threshold t is said to be *secure* if the view of the adversary that corrupts up to t players does not enable him to compute decryptions or signatures on his own. A threshold scheme is said to be *robust* if, no matter what the corrupted t players do, the remaining (i.e. honest) players still output a valid decryption or signature. (For formal definitions of security and robustness, see previous work [SG98,CG99,CGJ$^+$99b].)

A standard technique of proving security of a threshold cryptosystem (or a signature scheme) is to exhibit a *simulation* algorithm which, without access to any secret information but with an oracle access to the single-server realization of the underlying cryptosystem, furnishes the adversary with the correct view of the execution of the threshold protocol. Thus, by exhibiting such simulator, we *reduce* the security of the threshold version of a cryptosystem to the security of its single-server counterpart.

A corresponding standard technique for proving robustness of a threshold scheme is to exhibit a knowledge *extractor* which plays the part of the honest players in the protocol, and in case the adversary succeeds in inducing the honest players into producing an invalid output, it extracts from the adversary's behavior a solution to some hard problem. Thus again, by exhibiting such extractor, we *reduce* the robustness of our threshold protocol to some standard hardness assumption.

Previous adaptively secure solutions. The task of strengthening statically-secure protocols to handle an adaptive adversary contains a following difficulty: To compute an instance of a certain function robustly (we abstract from whether the function is a signature or a decryption), say an exponentiation function $A{=}m^a$ on instance m, where a is secret-shared, the players must publish some partial results of this function, say values $A_i{=}m^{\alpha_i}$ where α_i's are the polynomial shares of a. In the static model, since the group of corrupted players is fixed, without knowing a, on the input $A{=}m^a$ received from the function oracle (see the *definitions* paragraph above), the simulator can produce the view of the protocol that outputs A by picking the shares of the corrupted players and using them to interpolate values A_i of the honest players. However, in the adaptive model, such simulation fails because the simulator cannot publish A_i's for the honest players and then be able to open the values α_i s.t. $m^{\alpha_i}{=}A_i$ for any t-sized group of players that the adaptive adversary chooses to corrupt: That would imply the knowledge of more than t shares α_i, and hence the knowledge of a.

Recent adaptively secure protocols [CGJ+99,FMY99a-b] have overcome this difficulty with the following ideas: i) Value A can be reconstructed if every player publishes $A_i = m^{a_i}$ where a_i is its *additive* share of a, i.e. $\sum a_i{=}a$; ii) Robustness, previously guaranteed via "interpolation in the exponent" of values A_i, is achieved via generation of Pedersen's commitments $g^{a_i}h^{\hat{a}_i}$ along each share a_i, and with zero-knowledge proofs that show that m^{a_i} corresponds to the committed value. Because the shares are additive, the simulator can reveal a consistent internal state a_i for *all but one* of the players it controls. If that single *inconsistent* player is corrupted, which happens with at most $1/2$ probability, the simulation is rewound to the beginning of this instance of the function application. Thus the simulation of a single instance succeeds after expected two trials. However, since such rewinding must be constrained to within the single instance of the function application (see the overview discussion of Canetti et al. [CGJ+99a]), the additive shares a_i used in this protocol must be erased (resharing must be performed to enable the application of the function on a new instance), so that the simulator will know again the proper internal state of that player: He simply no longer needs to show the information that he cannot produce.

Concurrent adaptive security with committed proofs. Our first observation about the above reasoning is that there might be *no* inconsistent player during the simulation at all, if the "compromising" share a_i can be erased *before* the partial result A_i is published. Since there would be no inconsistent players, the simulator would never have to rewind, and hence *concurrent* executions of such threshold protocol can be simulated and thus proven secure. However, how can we achieve robustness if a player is to erase its share a_i before publishing A_i? We show that it is indeed possible by devising a novel tool of a *committed* zero-knowledge proof (see Sec. 3), where a statement that needs to be proven, e.g. "A_i and $g^{a_i}h^{\hat{a}_i}$ contain the same value a_i", is revealed only after the proof ends. In particular, it can be revealed after the witness a_i needed to prove the above statement is erased. This committed proof technique can thus be applied to transform, with negligible increase in communication complexity, the adaptive DSS and RSA solutions [CGJ+99,FMY99a-b], as well as other protocols like threshold ElGamal, to concurrently secure adaptive solutions.

We further observe that by providing robustness while eliminating all inconsistent players in the above way, the committed proof technique can actually transform, in the erasure-enabled setting, a very general class of *statically* secure threshold protocols into adaptively and concurrently secure ones (see Lysyanskaya [Lys00] for more discussion). In Section 6 we exemplify the generality of these techniques with a threshold Cramer-Shoup cryptosystem.

Erasure-free adaptive security with persistently inconsistent players. Our second observation is that in the above simulation [CGJ+99,FMY99a-b] a random inconsistent player need not be picked in a simulation of *each instance* of the function application protocol. Instead, it can pick some player at the beginning of the simulation process, and use that player as a single *persistently inconsistent* player in a simulation of each instance of the function application. If that player is ever corrupted, the simulation fails, but since that happens only with at most $1/2$ probability, such simulation still establishes a reduction from the security of the threshold protocol to the security of the underlying cryptosystem or signature scheme. If we can show that indeed this single player is the only player whose internal state held by the simulator is inconsistent with the adversary's view of the protocol, then our protocols do not have to resort to erasure, and hence they are secure in the erasure-free model.

We achieve this goal in two steps: First we remove the need to erase on the protocol level, by which we mean that the resulting scheme is secure in erasure-free model if it is implemented over *secure channels*. We do this, in general, by using additive sharing instead of polynomial sharing throughout the threshold protocols. Secondly, for the threshold protocols that need secure channels, we need to devise an encryption that implements the secure channels abstraction in the adaptive erasure-free model. This is an intriguing and non-trivial task, and the solution of *non-committing encryption for the receiver* which we provide in Section 4 is better than the available solutions [CFGN96,Bea97] of general non-committing encryption because it does not introduce any non-negligible communication overhead. The reason why non-committing encryption for the receiver only is sufficient is because not only is our simulator able to reveal, at the time of corruption, the history of computation of all players it controls except the persistently inconsistent one, but he already knows the values of all messages sent by these players at the time the messages are sent. These techniques yield efficient adaptive non-erasing protocols for DSS, RSA, and ElGamal (additionally, our methods lead to a dramatic reduction in the cost of adaptive RSA [JL00]). In Section 7 we exemplify them with a threshold Cramer-Shoup cryptosystem.

Finally, we remark that since the simulators in our erasure-free protocols are also non-rewinding (although they have 1/2 probability of faiure), a concurrent execution of any number of instances of such protocols is secure if, for example, they are executed by dedicated players (see [JL00] for more details).

3 Adaptive, concurrent security via committed proofs

In this section, we present the notion of a committed proof [Lys00], which is a zero-knowledge proof that is carried out *in a committed form.* The verifier does not learn the statement that is being proven until the very last round of the protocol. As discussed in section 2, this technique gives a general tool that transforms statically secure threshold protocols to adaptively secure ones, with the additional property that their security is preserved under concurrent composition.

Suppose we are given a following three-step public-coin honest-verifier zero-knowledge proof of knowledge system Z [BG92] for language L:

1. The proof system has perfect completeness and soundness $2^{-\Omega(k)}$.
2. The prover's input is $x \in L$, a witness w, and some randomness r.
3. The random coins R are tossed after the prover issues the first message.
4. Algorithms $P_1(x,w,r)$, and $P_2(x,w,r,R)$ generate the first and second messages of the prover.
5. The verifier runs algorithm $Ver(x,m_1,R,m_2)$ to determine whether to accept or reject.
6. The simulator algorithm SIM used for proving the zero-knowledge property of Z, has the property that for all inputs $R \in \{0,1\}^k$, it generates an accepting transcript (m_1,R,m_2) indistinguishable from a transcript of a conversation with the real prover.
7. The knowledge extractor algorithm KE for Z has the property that, for some constant c, on input $(x,m_1,R,R',\ldots,R^{(c)},m_2,m_2',\ldots,m_2^{(c)})$ such that $R \neq R' \neq \ldots \neq R^{(c)}$ and Ver accepts all transcripts (x,m_1,R,m_2), (x,m_1,R',m_2'), $\ldots$, $(x,m_1,R^{(c)},m_2^{(c)})$, KE outputs a witness w with probability $1-neg(k)$.

Such proof systems exist for all languages in NP, by a witness-preserving reduction to Hamiltonian cycles [Gol95]. In particular, for proving knowledge or equality of discrete logarithms or representations, such proof systems have perfect simulations and are well-studied and efficient [Bra99,Cam98].

Suppose that x for which the prover is demonstrating membership in L is unknown to the verifier. However, the verifier knows the distribution $\mathcal{D}$ from which x has been sampled. Moreover, $\mathcal{D}$ has the property that there is an efficiently samplable joint distribution $(\mathcal{W}, \mathcal{D})$ from which pairs (w,x) are sampled, such that w is a witness for the statement $x \in L$. For example, x can be a tuple $(G_q,\ g,\ h,\ y)$ and statement $x \in L$ means that y is an element in G_q that can be represented in bases g and h. When we sample $\mathcal{D}$, we can first generate a random $\alpha,\beta \in \mathbb{Z}_q$, then and then set $w = (\alpha,\beta)$, and $y = g^\alpha h^\beta$.

Suppose we are given a *trapdoor* commitment scheme, i.e. a commitment scheme that has the property that for any instance of the commitment scheme, there exists a trapdoor σ the knowledge of which enables to open any commitment to an arbitrary value within some given domain.

For example, consider Pedersen commitment: an instance is a group G_q of order q in which the discrete logarithm problem is hard, with generators g and h and a collision-resistant hash function $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$. The trapdoor $\sigma = \log_g h$. To commit to x, choose a random r and output $g^{\mathcal{H}(x)}h^r$. To open the commitment, reveal x and r. If σ is known, it is easy to see that a commitment can be opened to any x. Note that if we are not given a collision-resistant hash function, then the prover can still commit to his input x and the first message of the proof, but this commitment will have to use some special encoding of x and will be larger.

How can we create a simulator such that σ is known to it? In multi-party systems, we can have an instance of the commitment scheme generated as part of the set-up for the system; then it will follow from the properties of multi-party computation that a simulator will know σ. We will discuss such a protocol in section 5.1. In two-party protocols, σ can be a value known to the verifier, but not the prover; the simulator with black-box access to the verifier will then have to *extract* σ from the verifier.

Using trapdoor commitments, the prover can execute the proof *without revealing x to the verifier until the very end of the proof.* Consider the protocol in figure 1 between a prover and a verifier. The protocol uses Pedersen commitment, but any trapdoor commitment can be used instead.

Common inputs: (G_q, g, h): an instance of Pedersen commitment.
Prover's inputs: statement $x \in \mathcal{D}$, witness w, random input r.
Verifier's goal: obtain x s.t. prover knows a witness to "$x \in L$."

$P \longrightarrow V$ Prover computes $m_1 = P_1(x, w, r)$, chooses random r_1 and sends $M_1 = g^{\mathcal{H}(x,m_1)}h^{r_1}$.
$P \longleftarrow V$ Verifier tosses random coins R and sends them to the prover.
$P \longrightarrow V$ Prover computes $m_2 = P_2(x, w, r, R)$, chooses random r_2 and sends $M_2 = g^{\mathcal{H}(m_2)}h^{r_2}$. Prover erases w.
$P \longrightarrow V$ Prover sends x, m_1, m_2, r_1, r_2, i.e. opens commitments M_1, M_2.
Acceptance: The verifier accepts if M_1 is a valid commitment to x and m_1, M_2 is a valid commitment to m_2, and $Ver(x, m_1, R, m_2)$ accepts.

Fig. 1. Committed proof

Note (Completeness): We get completeness for free from proof system Z.

Lemma 1. (Zero-knowledge) *This protocol is zero-knowledge for any verifier.*

Proof: The lemma follows from the fact that for a simulator that knows $\log_g h$, the commitments M_1 and M_2 are not binding, and so the simulator can reveal x, message m_1 and response m_2 in the very end, when it already knows the challenge R, by property 6 of proof system Z. □

Note: Notice that the original proof system Z was zero-knowledge for the public-coin model only, while the proof system we obtain is zero-knowledge for *any* verifier. (We achieve this because of a preprocessing step that generates h.)

Lemma 2. (Concurrent composition) *This protocol remains secure when executed concurrently (i.e. with an arbitrary interleaving of steps) with arbitrarily many invocations of itself or of any other concurrently composable protocols.*

Proof: The lemma follows from the fact that the above simulator that exhibits the zero-knowledge property does not need to rewind the verifier. □

Lemma 3. (Soundness and knowledge extraction) *If the discrete logarithm problem is hard, and the hash function $\mathcal{H} : \{0,1\}^* \to {\mathbb{Z}_q}^*$ is collision-resistant, then for this protocol there exists a polynomial-time knowledge extractor such that if the verifier accepts with non-negligible probability, then with probability $1 - neg(k)$ the knowledge extractor learns the witness w for x that the prover possesses.*

Proof: We will exhibit a knowledge extractor which, with black-box access to the prover that induces the verifier to accept with non-negligible probability, either extracts a witness for x or computes the discrete logarithm of h to the base g, or finds a collision in $\mathcal{H}$. Clearly this is sufficient to prove the lemma.

The extractor runs the prover and obtains the x, as well as m_1, R, m_2 and M_1, r_1, M_2, r_2. Now the extractor rewinds the prover to step 3 of the protocol and issues a challenge $R' \neq R$. Running the protocol to the end allows the verifier to obtain x', as well as m_1', m_2', r_1', r_2' and M_2'. Note that since the prover replies with non-negligible probability, with enough rewindings, we will get as many replies from him as the knowledge extractor KE of proof system Z may need.

Suppose $x \neq x'$. Then either $x = \mathcal{H}(x, m_1) \neq \mathcal{H}(x', m_1') = x'$ or we have found a collision in the hash function. If the latter, we have the desired contradiction. Otherwise, $g^x h^{r_1} = M_1 = g^{x'} h^{r_1'}$, and so we can compute $\log_g h$.

Now suppose $x = x'$. Then, by the same argument as above, $m_1 = m_1'$ or we find a collision or compute discrete log. Then since m_2 is a valid response to challenge R and so is m_2' to challenge R', it follows from the fact that Z is a proof of knowledge that we can extract a witness for x by using KE. □

Finally, lemma 4 below is the key to why a committed proof is instrumental for designing protocols that are secure against the adaptive adversary. It captures the counter-intuitive fact that the prover can be attacked in the middle of the proof, but the adversary still learns nothing, i.e. the zero-knowledge property of the whole game is retained! The only condition required is that the distribution $(\mathcal{W}, \mathcal{D})$ that captures the adversary's *a priori* information about the distribution that x and witness w come from, be efficiently samplable.

Lemma 4. (Security against corruption) *If the prover is corrupted by the adversary in the middle of the proof,everything that the adversary learns can be accomplished either by revealing x, or by sampling $(\mathcal{W}, \mathcal{D})$.*

Proof: We prove the claim by exhibiting a simulator $\mathcal{S}$ which generates the adversary's view of the corruption. Suppose the adversary decides to corrupt the prover just before the end of step 3. Then $\mathcal{S}$ samples $(\mathcal{W}, \mathcal{D})$ and obtains a witness w' for an x'. $\mathcal{S}$ the generates a random r and, using trapdoor $\sigma = \log_g h$ computes $m_1' = P_1(x, w, r)$ and r_1' such that $M_1 = g^{\mathcal{H}(x', m_1')} h^{r_1'}$, as well as $m_2' = P_2(x, w, r, R)$ and r_2' such that $M_2 = g^{\mathcal{H}(m_2')} h^{r_2'}$. Reveal w', x', r, r_1', r_2' to the adversary. These values are distributed correctly since w' and x' come from distribution $(\mathcal{W}, \mathcal{D})$ and r, r_1', r_2' are all random values.

Suppose the adversary decides to corrupt the prover at some step before the end of step 3. Then it is clear that $\mathcal{S}$ will just have to reveal a subset of the values above (depending on whether M_1 and M_2 have been issued yet).

Suppose the adversary corrupts the prover after the end of step 3, i.e. after w was erased. Since w is erased, the adversary learns nothing more than what the verifier can learn. Thus, $\mathcal{S}$ just runs the simulator we have constructed for proving the zero-knowledge property. □

As we will see in section 6, this property of a committed proof allows us to create a perfect and never failing simulation of the adversary's view, which implies full concurrency of the erasure-enabled threshold cryptosystems we propose.

4 Implementing secure channels without erasures

In erasure-enabled adaptive threshold cryptosystems (for example our threshold Cramer-Shoup of Sec. 6) we can assume secret communication between players because they can be implemented in that model with an inexpensive technique due to Beaver and Haber [BH92]. However, if erasures are not allowed, implementing secure channels is more complicated. The problem arises because the adversary can tap all the channels and see all the ciphertexts passed between players. When the adaptive adversary corrupts a party, he expects to see cleartexts that correspond to the ciphertexts he has seen. Thus the adaptive adversary can potentially open any generated ciphertext. When instead of the honest players, we have a simulator attempting to simulate the adversary's view (recall that such simulator is needed to prove security), we cannot easily argue why the adversary does not learn anything from, paradoxingly, the ciphertexts that he does *not* get to open. This subtle problem, known as *selective decommitment problem* (see Dwork et al. [DNRS99]), arises, from our inability to reduce an adversary that *does* learn something from such view to semantic security of encryption. This problem can be solved with a non-committing encryption, i.e. an encryption with an additional property that the ciphertext-looking messages sent by the simulator can be opened as any cleartexts, and hence contain no information.

A general solution to this problem, due to Canetti et al. [CFGN96], requires $O(k^2)$ communication for secure transmission of a single bit, where k is the security parameter. A less expensive technique under the decisional Diffie-Hellman requires $O(k)$ overhead and is due to Beaver [Bea97].

We present a conceptually simpler but less general encryption scheme E which, under the DDH assumption, is non-committing for the receiver only [JL00]. Such encryption is custom-made for the *persistently inconsistent player* paradigm. Namely, a simulator who sends the ciphertext-looking messages on behalf of the inconsistent player is able to open them freely if the adversary attacks any receiver of these messages, i.e. anybody but the inconsistent player. Since our simulation assumes that the adversary never corrupts that player anyway (which gives us 1/2 probability of success), such encryption is good enough for simulatability of our protocols. The non-committing encryption we propose has only negligible communication overhead.

E is a non-committing encryption scheme in the following sense: On the one hand, any properly encrypted message has a unique decryption. On the other, there is a procedure which, given a sender's key and some trapdoor σ, can produce special type of *invalid* ciphertexts, which, for any $a \in \mathbb{Z}_q$, can be opened as an encryption of $m = g^a$. This is achieved because there are q possible secret keys that this procedure can reveal. Moreover, under DDH, it is impossible to distinguish the regular ciphertexts and the invalid ones produced by this special procedure. The ideas we use to implement this encryption E are similar to those of Cramer and Shoup [CS98].

Common system parameters: Group G_q, generators g and h.
Bob selects: $x, y \in \mathbb{Z}_q$, **Bob sends to Alice:** $P = g^x h^y$.
Alice: To transmit message $m \in G_q$ to Bob, Alice chooses $r \in Z_q$ and sends $A = g^r$, $B = h^r$, $C = P^r m$ to Bob.
Bob: Computes $m = C/(A^x B^y)$.

Fig. 2. Non-committing encryption scheme E

Lemma 5. *Under DDH, E is non-committing for the receiver.*

Proof: Suppose that Alice (the sender) and Bob (the receiver) are on the same side and both know $\sigma = \log_g h$ and $z = \log_g P$. Then they can compute an *invalid* ciphertext as follows: Pick $r_1 \neq r_2 \neq r_3$ at random, and let $A^* = g^{r_1}$, $B^* = g^{r_2}$, $C^* = g^{r_3}$. (A^*, B^*, C^*) is not a valid ciphertext because $r_1 \neq r_2$. If Bob is infiltrated, then for any $m_a = g^a$, he can claim that this triple is an encryption of m_a, by showing a secret key (x^*, y^*) such that the decryption algorithm outputs m_a. He can do that by solving a system of linear equations: $x^* + y^*\sigma = z \bmod q$ and $r_3 = r_1 x^* + r_2 y^* \sigma + a \bmod q$. If $r_1 \neq r_2$ this system must have a solution. Therefore, as long as a, σ and z are known to Alice and Bob, they are not committed to the plaintext.

We must now show that whether the ciphertext sent is valid or invalid as above the view of the adversary who is observing the conversation and may infiltrate Bob remains the same. Let us call the distribution that produces the tuples (P, A^*, B^*, C^*) of the invalid form, $\mathcal{E}^*(G_q, g, h)$. By $\mathcal{E}(G_q, g, h)$, we will denote the distribution that produces the tuples (P, A, B, C) where (A, B, C) is a valid ciphertext under key P. We will now show that $\mathcal{E}$ and $\mathcal{E}^*$ are computationally indistinguishable under the DDH assumption.

Suppose a DDH instance (g, u, v, w) is given. Our goal is to decide whether it was sampled according to distribution $\mathcal{D} = \{g, g^s, g^t, g^{st}\}_{\{s,t\}}$ or according to distribution $\mathcal{D}^* = \{g, g^s, g^t, g^z\}_{\{s,t,z\}}$. Create the common information for the encryption scheme as follows: Choose values α and β such that $h = g^\alpha u^\beta$. Choose x and y and create $P = g^x h^y$. Choose some random a, b, r. Send (A, B, C) where $A = (g^a v^b)^r$, $B = A^\alpha ((u^a w^b)^r)^\beta$, and $C = A^x B^y m$. Note that if $\log_g w = \log_g u \log_g v$ (i.e. the DDH instance is from $\mathcal{D}$), then the view the adversary gets is from distribution $\mathcal{E}$; otherwise the adversary's view is from distribution $\mathcal{E}^*$. Thus, the adversary that distinguishes between $\mathcal{E}$ and $\mathcal{E}^*$ can be used to distinguish between $\mathcal{D}$ and $\mathcal{D}^*$. Therefore, under DDH, no such polynomial-time adversary exists. $\square$

Lemma 6. *If a multi-party protocol is secure against the adaptive adversary in the secure channel erasure-free model, and the simulator algorithm SIM^* used to prove security produces a perfect simulation and is such that all but a constant number of players controlled by this simulator (i.e. the inconsistent players) follow the protocol exactly, and all messages sent by all honest players can be prepared by the simulator at send-time such that (1) the inconsistent player's messages are selected uniformly at random and (2) other players' messages are distributed correctly in full consistency with whatever the simulator will open as this player's internal state, then using encryption E results in a secure multi-party protocol in insecure channels (under the DDH assumption).*

Proof Sketch: First we notice that, since the messages of the honest and consistent players are known to SIM^*, the erasure-free simulator SIM that we need to construct just uses E to encrypt the right message from them all the time. Second, we note that since the messages of the inconsistent player can also be prepared at send-time, the simulator can prepare sender's key, receiver's key, ciphertext tuples that would decrypt to these messages.

Next, we notice that if SIM uses scheme $\mathcal{E}^*$ for the inconsistent players, then, whether the simulator knows the secret inputs and follows the protocol (call that $View_1$) or simulates it as SIM^* would (call that $View_2$), the adversary sees no difference in the view because the "ciphertexts" produced by $\mathcal{E}^*$ are independent of the messages sent on the part of the sender. Now, assume that the simulator knows the players' inputs and follows the protocol, but embeds an instance of DDH into the common system parameter h, as described in lemma 5, into the ciphertext-looking messages produced on the part of the inconsistent players. This construction creates information-theoretically independent samples of $\mathcal{E}$ or $\mathcal{E}^*$ based on the same instance of the DDH (call the view of the first distribution $View_3$, and note that the second view is $View_1$ discussed above). Therefore, the adversary that differentiates these two distributions can be used to solve the DDH. Hence $View_3$ is the view of the protocol over the insecure channels, and $View_2$ is a view of a simulation, this protocol is secure. □

We note that this implementation of secure channels can only work for a special class of multi-party protocols, namely, the ones that satisfy the conditions of lemma 6. Thus, although it does not replace Beaver's elegant scheme in general, it allows us to create efficient erasure-free adaptive protocols for many schemes that are important in practice, like RSA, DSS, ElGamal, and Cramer-Shoup.

5 Common building blocks

5.1 Set-up: Generating an Instance of a Trapdoor Commitment

Our protocols rely heavily on a discrete-log based trapdoor commitment scheme due to Pedersen: On instance (p, q, g, h), where $h \in G_q$, a commitment to $x \in \mathbb{Z}_q$ is $C = g^x h^{\hat{x}}$, where $\hat{x}$ is picked at random in $\mathbb{Z}_q$. The value h that defines the commitment instance is generated jointly once and for all at the beginning of our protocols in such a way so that (1) the simulator can learn the trapdoor $\log_g h$ of the chosen commitment; and (2) the simulator can embed another instance of the discrete log problem into the generated commitment by learning the representation of h in bases g, $\tilde{g}$ of its choice. Option i) is used for proving secrecy, when knowledge of the trapdoor enables the simulator to always open the commitments of the players it controls in the way it chooses, which leads to efficient simulation of the protocols. Option ii) is used to prove robustness: If the adversary cheats in protocols that follow, the simulator can use such adversary to break an instance of the hard problem embedded in the trapdoor. When secure channels are present, h can be obtained by using general techniques of multi-party computation [BGW88,CDD+99]. When secure channel are not there, and implementing them by erasure is not an option, we can use another protocol, where each player generates his share h_i of h, and then all players, in parallel, prove knowledge of $\log_g h_i$ to each other. This is in some respect similar to the solution of Frankel et al. [FMY99a-b]. Please see Jarecki and Lysyanskaya [JL00] for the details.

5.2 Joint Random VSS and Distributed Coinflip

In Figure 3, we include the well-known protocol Joint-RVSS [Ped91,GJKR99] for joint verifiable sharing of a random secret, which is a basic building block of our protocols. We give it here anew using notation that is useful for the presentation of the protocols that follow.

Protocol: (on inputs group G_q, generators g, h)
1. Each player P_i performs a Pedersen VSS of a random value a_i:
(a) P_i picks t-deg. polynomials $f_{a_i}(z) = \sum_{k=0}^{t} c_{ik} z^k$, $f_{\hat{a}_i}(z) = \sum_{k=0}^{t} \hat{c}_{ik} z^k$
Let $a_i = f_{a_i}(0)$ and $\hat{a}_i = f_{\hat{a}_i}(0)$ be the values shared by these polynomials
P_i broadcasts $C_{ik} = g^{c_{ik}} h^{\hat{c}_{ik}}$ for $k = 0..t$. Set $F_{a_i}(z) = \prod_{k=0}^{t} (C_{ik})^{z^k}$
P_i sends to P_j shares $\alpha_{ij} = f_{a_i}(j)$, $\hat{\alpha}_{ij} = f_{\hat{a}_i}(j)$ for each $j = 1..n$
(b) Each P_j verifies if $g^{\alpha_{ij}} h^{\hat{\alpha}_{ji}} = F_{a_i}(j)$ for $i = 1..n$
If the check fails for any i, P_j broadcasts a *complaint* against P_i
(c) If P_j complained against P_i, P_i broadcasts α_{ij}, $\hat{\alpha}_{ij}$. Everyone verifies it.
If P_i fails this test or receives more than t complains, exclude P_i from $Qual$
2. P_i sets his polynomial share of the generated secret a as
$\alpha_i = \sum_{P_j \in Qual} \alpha_{ji}$, and their associated randomness as $\hat{\alpha}_i = \sum_{P_j \in Qual} \hat{\alpha}_{ji}$

We label the data structure created by this protocol as $\mathsf{RVSS\text{-}data}_{t,g,h}[a]$:
Secret Information of each player P_i: (well-defined for $P_i \in Qual$)
– $a_i, \hat{a}_i$ his additive shares of the secret and its associated randomness
– $f_{a_i}, f_{\hat{a}_i}$ t-degree polynomials he used in sharing his additive share
– $\alpha_i, \hat{\alpha}_i$ his polynomial share of the secret and its associated randomness
– $\alpha_{ji}, \hat{\alpha}_{ji}$ his polynomial shares (and assoc. rand.) of $f_{a_j}, f_{\hat{a}_j}$ for $j = 1..n$
Public Information:
– the set $Qual \subseteq \{P_1, .., P_n\}$
– verification function $F_a : \mathbb{Z}_q \rightarrow \mathbb{Z}_p{}^*$ (see the implicit information below)
– verification functions $F_{a_i}(z) = g^{f_{a_i}(z)} h^{f_{\hat{a}_i}(z)}$ for $P_i \in Qual$
Secret Information Defined *Implicitly* (not stored by any player):
– secret sharing t-degree polynomials $f_a(z), f_{\hat{a}}(z)$ s.t. $\alpha_i = f_a(i)$, $\hat{\alpha}_i = f_{\hat{a}}(i)$,
$f_a(z) = \sum_{P_i \in Qual} f_{a_i}(z)$, $f_{\hat{a}}(z) = \sum_{P_i \in Qual} f_{\hat{a}_i}(z)$, and $F_a(z) = g^{f_a(z)} h^{f_{\hat{a}}(z)}$
– secret shared value $a = f_a(0)$ and its associated randomness $\hat{a} = f_{\hat{a}}(0)$

Fig. 3. Joint-RVSS creates a sharing RVSS-data[a] of random secret $a \in \mathbb{Z}_q$

Notation: We say that players generate $\mathsf{RVSS\text{-}data}_{t,g,h}[a]$ if they execute this protocol with generators g, h and polynomials of degree t. We index the data produced with labels a, α, using the associated Greek letter for polynomial shares.

One use of Joint-RVSS is in a distributed coinflip protocol (Fig.4), whose security properties are formalized in Lemma 7. This lemma is useful also for other uses of Joint-RVSS, where unlike in the coinflip protocol, the generated secret is not explicitely reconstructed.

Lemma 7. *In secure channels model, the distributed coinflip protocol of Fig. 4 (1) does not use erasures and (2) simulator* SIM *simulates it without rewinding.*

Proof: The simulator for the security proof is contained in figure 4. The simulator knows $\log_g h$, thus it need not decide on a_i's for players P_i it controls until it learns a_j for each player P_j that the adversary controls. (Note that the simulator can determine the adversary's value a_j by interpolating $f_{a_j}(i)$.) After that, the simulator assigns values a_i to the players in such a way that $\sum_{P_i \in Qual} a_i = a^*$. □

Note: If the simulator is allowed to have one player $P_i \in Qual$ whose internal state is inconsistent, then it can decide on the values a_k in advance for all $P_k \neq P_i$, and only leave a_i undefined until it is able to set $a_i = a^* - \sum_{P_k \in Qual, k \neq i} a_k$. This observation will be useful for erasure-free protocols.

Protocol: (on inputs group G_q, generators g, h)
1. Players generate RVSS-data[a] (i.e. perform Joint-RVSS, Fig.3)
2. Each $P_i \in Qual$ broadcasts his additive shares $a_i, \hat{a}_i$
3. For $P_i \in Qual$ s.t. $g^{a_i} h^{\hat{a}_i} \neq F_{a_i}(0)$, the players reconstruct P_i's additive share a_i by broadcasting their shares $\alpha_{ij}, \hat{\alpha}_{ij}$ and verifying them with F_{a_i}
4. A public random value a is reconstructed as $a = \sum_{P_i \in Qual} a_i$

Simulation: (on SIM's inputs G_q, g, h and $\sigma = \log_g h$)
1. SIM performs Joint-RVSS on the part of the honest players
2. SIM receives random $a^* \in \mathbb{Z}_q$. For some P_i among the players it controls: SIM broadcasts $a_i^* = a^* - \sum_{P_j \in Qual \setminus \{P_i\}} a_j$, $\hat{a}_i^*$ s.t. $a_i + \sigma \hat{a}_i = a_i^* + \sigma \hat{a}_i^*$
   For all other players P_j it controls, SIM broadcasts correct values $a_j, \hat{a}_j$
3. SIM performs Step 3 on the part of the honest players
4. Note that the public random value is reconstructed as a^*

Fig. 4. Erasure-Free Distributed Coinflip Protocol using Joint-RVSS

5.3 Simultaneous Zero-Knowledge Proofs of Knowledge

Our adaptive protocols, following the protocols of Canetti et al. [CGJ+99a], use *simultaneous* zero-knowledge proofs of knowledge to enable robustness efficiently. We describe this technique here in full generality.

Consider any honest-verifier public-coin zero-knowledge proof of knowledge system (ZKPK) [BG92]. Say that the prover shows knowledge of witness w of a public relation $A = (y, x)$ for some value y. Let (p, q, g) be a discrete-log instance and assume that the random coins in the proof system are picked in $\mathbb{Z}_q$. Assume that the simulator that exhibits the zero-knowledge property proceeds by first choosing any value for the random coin and then generating the rest of the proof transcript, and that it has zero probability of failure. Three-round ZKPKs of this form exist for, in particular, proving knowledge of discrete logarithm, i.e. $A = \{g^x, x\}$ (e.g. Schnorr's protocol [Sch91]), or knowledge of representations, e.g $A = \{(g, h, g^x h^{\hat{x}}), (x, \hat{x})\}$ (see the work of Brands [Bra99] or Camenisch [Cam98] and the references therein). In a simultaneous proof using a three-round ZKPK, each player P_i proves knowledge of its witnesses w_i for some statement (y_i, w_i) in A in parallel, by executing the steps of the prover as in the original ZKPK protocol, while for the verifier's part, they all use a single common public coin generated with a distributed coinflip protocol. In our protocols, such simultaneous proof is preceeded by h-generation and the coinflip is implemented with the protocol in Fig.4. This method generalizes to ZKPK protocols with any number of rounds: Every time a public coin is needed, it is picked via a distributed coinflip.

The following lemma is purely technical, but it isolates a convenient property of the simultaneous proof that allow us to concisely argue the security of the protocols that use it as a building block.

Lemma 8. *In the secure channels model, the simultaneous proof protocol has the following two properties: (1) It can be simulated without rewinding as long as the simulator has a consistent internal state for every player the adversary corrupts; (2) There is a simulator that can extract all the witnesses from the players controlled by the adversary.*

See Jarecki and Lysyanskaya [JL00] for the proof. From the lemma above and lemma 4 we immediately get:

Corollary 1. *In the erasure-enabled model, if the ZKPK proof used in the above simultaneous proof protocol is a committed proof of Fig.1, this protocol can be succesfully simulated without rewinding even if the simulator does not know any witnesses to the statements it reveals for the players it controls.*

Lemma 8 also implies a corollary useful for our erasure-free protocols:

Corollary 2. *In the secure channels erasure-free model, the simultaneous proof protocol can be simulated if the simulator does not know the witnesses for a constant number of players it controls, as long as these players are not corrupted by the adversary.*

5.4 Shared Exponentiation Protocol

Another useful building block of our threshold cryptosystems is a protocol that computes m^a for any input element $m \in G_q$ if value $a \in \mathbb{Z}_q$ is secret-shared with $\mathsf{RVSS\text{-}data}[a]$. This protocol has two variants, an "additive" and "polynomial" exponentiation (Figs. 5 and 6), which refers to the two methods of extracting value m^a from the sharing $\mathsf{RVSS\text{-}data}[a]$ of a: Every player P_i broadcasts either value m^{a_i} for its additive share a_i, or value m^{α_i} for its polynomial share α_i.

The additive exponentiation protocol, which generalizes and removes erasure from the distributed key generation protocol of [CGJ$^+$99a], is a basis of the key generation for our threshold Cramer-Shoup cryptosystems, and it is used in our threshold Cramer-Shoup decryption in the erasure-free setting. The polynomial exponentiation is used in our concurrent erasure-enabled Cramer-Shoup decryption. Since the polynomial exponentiation protocol erases the polynomial shares α_i of a at the end, in that model we must always use a one-time randomization of the polynomial secret-sharing of a as inputs to this protocol. We omit the proofs of the two lemmas below and send the reader to Jarecki and Lysyanskaya [JL00] and Lysyanskaya [Lys00] for them.

Input: $m \in G_q$, secret sharing RVSS-data$[a]$, $g, h \in G_q$

1. Each P_i broadcasts $A_i = m^{a_i}$
2. With a simultaneous proof of Sec. 5.3, using ZKPK proof of equality of representation, each P_i proves knowledge of (equal) representation of m_i in bases $m, 1$ and $F_{a_i}(0)$ in bases g, h.
   If some P_i fails, a_i and $A_i = m^{a_i}$ are reconstructed publicly using F_{a_i}
3. Everyone computes $m^a = \prod_{i=1}^n A_i$

Fig. 5. Erasure-Free Additive Exponentiation with RVSS-data$[a]$

Lemma 9. *In the secure channels erasure-free model, as long as the adversary does not corrupt the designated persistently inconsistent player, the additive exponentiation protocol can be simulated such that (1) for all honest and consistent players, the simulator can provide correct messages they send at the time of sending and (2) for the honest inconsistent player, the simulator can provide messages such that if any t of them are revealed they look correct.*

Input: $m \in G_q$, secret sharing RVSS-data$[a]$, $g, h \in G_q$

1. With a simultaneous proof of Sec. 5.3, using *commited* ZKPK proof (Fig.1) of equality of representation, each P_i proves knowledge of (equal) representation of $A_i = m^{\alpha_i}$ in bases $m, 1$ and $F_a(i)$ in bases g, h.
   Note that at the end of the proof, value A_i is published and α_i erased.
2. Value m^a is interpolated in the exponent from A_i's that passed the proof

Fig. 6. Erasure-Enabled Polynomial Exponentiation with RVSS-data$[a]$

Lemma 10. *In the erasure-enabled model, the polynomial exponentiation protocol can be simulated without rewinding.*

6 Concurrent threshold Cramer-Shoup cryptosystem

The Cramer-Shoup cryptosystem. Recall the Cramer-Shoup [CS98] cryptosystem. The setting is as follows: a group G_q in which the decisional Diffie-Hellman problem is assumed to be hard, and a universal one-way family of hash functions $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q{}^*$ are given [BR97,Sho99a]. The secret key consists of five values, a, b, c, d, e, selected from $\mathbb{Z}_q{}^*$ uniformly at random. The public key consists of two random bases, $g_1, g_2 \in G_q$, such that the discrete logarithm that relates them is unknown, and the group elements $C = g_1^a g_2^b$, $D = g_1^c g_2^d$ and $W = g_1^e$. To encrypt a message m from a message space M (M is assumed to have an efficiently computable and invertible mapping into G_q, and so we write $m \in G_q$), Alice chooses $r \in \mathbb{Z}_q{}^*$ uniformly at random, computes $x = g_1^r$, $y = g_2^r$, $w = W^r m$, $\sigma = \mathcal{H}(x, y, w)$, and $v = C^r D^{r\sigma}$. The ciphertext is the 4-tuple (x, y, w, v). And now for decryption, we will use the

Canetti-Goldwasser method [CG99]: Bob selects uniformly at random $s \in \mathbb{Z}_q{}^*$ and outputs $w/(x^e(v/v')^s)$, where $v' = x^{a+c\sigma}y^{b+d\sigma}$. Recall that, under the assumption that the decisional Diffie-Hellman problem is hard, the Cramer-Shoup cryptosystem, as well as the Canetti-Goldwasser variant thereof, has been shown to be secure against adaptive chosen ciphertext attack which is the strongest notion of security known for public-key cryptosystems [CS98,Sho99b,CG99].

Key generation. In figure 7, we present the key generation protocol for the concurrent Cramer-Shoup cryptosystem. We assume that the group G_q with a generator g and the universal one-way hash function $\mathcal{H}$ have been generated already. Indeed we may allow one server to set up these parameters and have the others verify that his computation was performed correctly. We also assume that $h \in G_q$ was generated using correct h-generation protocol.

Input: G_q, g, h, $\mathcal{H}$
Goal: Generate the Cramer-Shoup public key (g_1, g_2, C, D, W).

1. Run the joint coinflip protocol and generate random bases g_1, g_2, h_1, h_2.
2. Run Joint-RVSS five times in parallel and obtain $\mathsf{RVSS\text{-}data}_{t,g_1,h_1}[a, c, e]$ and $\mathsf{RVSS\text{-}data}_{t,g_2,h_2}[b, d]$.
3. P_i performs, in parallel, committed simultaneous proofs of knowledge of repr. in bases g_1, g_2 of values $C_i = g_1^{a_i} g_2^{b_i}$, $D_i = g_1^{c_i} g_2^{d_i}$ and $W_i = g_1^{e_i}$;
   and repr. in bases h_1, h_2 of values $\hat{C}_i = h_1^{\hat{a}_i} h_2^{\hat{b}_i}$, $\hat{D}_i = h_1^{\hat{c}_i} h_2^{\hat{d}_i}$, and $\hat{W}_i = g_1^{\hat{e}_i}$;
   P_i erases f_{a_i}, f_{b_i}, f_{c_i}, f_{d_i}, f_{e_i} and $f_{\hat{a}_i}$, $f_{\hat{b}_i}$, $f_{\hat{c}_i}$, $f_{\hat{d}_i}$, $f_{\hat{e}_i}$;
   P_i opens the committed proofs.
4. Verify (1) validity of other players' proofs;
   and (2) for all $P_k \in Qual$, $C_k\hat{C}_k = F_{a_k}(0)F_{b_k}(0)$, $D_k\hat{D}_k = F_{c_k}(0)F_{d_k}(0)$,
   and $W_k\hat{W}_k = F_{e_k}(0)$.
   For any player who failed the test, reconstruct all his secrets using backup information stored in $\mathsf{RVSS\text{-}data}[a, b, c, d, e]$.
5. Compute the public key:
   $C = \prod_{P_i \in Qual} C_i$, $D = \prod_{P_i \in Qual} D_i$ and $W = \prod_{P_i \in Qual} W_i$.

Fig. 7. Erasure-Enabled Key Generation for Cramer-Shoup

See Lysyanskaya [Lys00] for proofs of security and robustness for this protocol. We note that the simulator for the security proof is easy to construct. The key step here is that we generate two auxiliary bases, h_1 and h_2, such that if this is a simulation, the simulator will get to know $\log_{g_1} h_1$ and $\log_{g_2} h_2$. As a result of this and of the committed proof technique, at no step of this protocol will the simulator be committed to a particular player's internal state (see lemma 1 and lemma 4). The additive share of the public key published at the end is non-committing to any current internal state either because it is distributed independently from any non-erased information that the adversary will ever have a chance to see.

We also note that if a corrupted player deviates from the protocol but still succeeds in carrying out the committed proof so that the honest players accept, then, since these proofs are proofs of knowledge of representation, we can exhibit an extractor which will compute two different representations of some value in two different bases, and will therefore solve the discrete logarithm problem.

Input: Values obtained from the key generation protocol.
Goal: Decrypt ciphertext (x, y, w, v).
Notation: In this protocol, indexed Latin letters (e.g. a_i) denote *polynomial* shares of the corresponding values. (Unlike the rest of this extended abstract where they denote additive shares.)

1. Run Joint-RVSS five times and obtain $\mathsf{RVSS\text{-}data}_{t,g,h}[s, r, p]$ and $\mathsf{RVSS\text{-}data}_{2t,g,h}[o, z, u]$.
2. P_i computes the following values:
   (a) $l_i = x^{a_i+c_i\sigma} y^{b_i+d_i\sigma} g^{r_i} = v_i g^{r_i}$, where $v_i = x^{a_i+c_i\sigma} y^{b_i+d_i\sigma}$.
   (b) $l'_i = g^{r_i} h^{p_i}$.
   (c) $l''_i = g^{r_i s_i} h^{p_i s_i + u_i} = (l'_i)^{s_i} h^{u_i}$.
   (d) $m_i = (l_i/v)^{s_i} g^{z_i} = (v_i/v)^{s_i} g^{r_i s_i} g^{z_i}$.
   (e) $m'_i = x^{e_i} g^{-r_i s_i} g^{o_i} g^{-z_i}$.
3. Prove in committed simultaneous form:
   (a) Eq. of repr. of l_i, $F_a(i)$, $F_b(i)$, $F_c(i)$, $F_d(i)$, $F_r(i)$ in bases $(x, x^\sigma, y, y^\sigma, g, 1, 1, 1, 1, 1)$, $(g_1, 1, 1, 1, 1, h_1, 1, 1, 1, 1)$, $(1, 1, g_2, 1, 1, 1, 1, h_2, 1, 1)$, $(1, g_1, 1, 1, 1, 1, h_1, 1, 1, 1)$, $(1, 1, 1, g_2, 1, 1, 1, 1, h_2, 1)$, $(1, 1, 1, 1, g, 1, 1, 1, 1, h)$, correspondingly.
   (b) Eq. of repr. of l'_i, $F_r(i)$, $F_p(i)$ in bases $(g, 1, h, 1)$, $(g, h, 1, 1)$, $(1, 1, g, h)$.
   (c) Eq. of repr. of l''_i, $F_s(i)$, $F_u(i)$ in bases $(l'_i, h, 1, 1)$, $(g, 1, h, 1)$, $(1, g, 1, h)$.
   (d) Eq. of repr. of m_i,$F_s(i)$, $F_z(i)$ in bases $((l_i/v), g, 1, 1)$, $(g, 1, h, 1)$, $(1, g, 1, h)$.
   (e) Eq. of repr. of m'_i, $F_e(i)$, l''_i, $F_o(i)$, $F_z(i)$ in bases $(x, g^{-1}, g, g^{-1}, 1, 1, 1, 1)$, $(g_1, 1, 1, 1, h_1, 1, 1, 1)$, $(1, g, 1, 1, 1, h, 1, 1)$, $(1, 1, g, 1, 1, 1, h, 1)$, $(1, 1, 1, g, 1, 1, 1, h)$.
4. Erase the one-time secrets generated in step 1.
5. Open the committed proofs and reveal l_i, l'_i, l''_i, m_i, and m'_i.
6. Verify the committed proofs of other players.
7. Set a players output share $O_i = m_i m'_i$. Determine the output O by Lagrange interpolation in the exponent; the resulting decryption is w/O.

Fig. 8. Erasure-enabled decryption for Cramer-Shoup

Decryption. In figure 8, we present the decryption protocol for the Cramer-Shoup cryptosystem. For full proofs of security and robustness of this protocol, see Lysyanskaya [Lys00].

Let us only show correctness of the decryption protocol in figure 8: if all the players behave as prescribed by the protocol, the output is valid decryption. To see this, let us look at the values O_i: $O_i = m_i m'_i = (v_i/v)^{s_i} g^{r_i s_i} g^{z_i} x^{e_i} g^{-r_i s_i} g^{o_i} g^{-z_i}$ $= (v_i/v)^{s_i} x^{e_i} g^{r_i s_i + z_i - r_i s_i + o_i - z_i} = (v_i/v)^{s_i} x^{e_i} g^{o_i}$ Since $o(i)$ is a degree $2t$ share of 0, the interpolation of these shares will yield $(v'/v)^{s(0)} u_1^z$, as in Canetti and Goldwasser [CG99].

The decryption protocol is secure because all the information that one sees before the committed proofs are opened does not commit the simulator to the internal state ofo any of the players (by lemma 4), and, since the simulator knows the values $\log_g h$, $\log_{g_1} h_1$ and $\log_{g_2} h_2$, the simulator can exhibit the internal state of any player at the adversary's request. The information revealed after the committed proof is information-theoretically independent of the internal state of a player who published this information, since by the time he publishes it, any secrets pertaining to it have been erased; and the whole process is perfect zero-knowledge by corollary 1. Therefore, owing to the committed proof technique we get a perfect simulation for the adversary's view. Robustness follows from lemma 3.

Key refresh. Notice that, using standard techniques [HJJ$^+$97], the above implementation of the threshold Cramer-Shoup cryptosystem can be made *proactive* i.e. secure against mobile attackers who, over time, lose control over some of the servers, but attack new ones.

Taking the decryption off-line. Note that, as in the Canetti-Goldwasser implementation [CG99], we can precompute and store the randomizers. When a ciphertext needs to be decrypted, a user can talk to each server individually and have each server, using committed proofs, prove to the user that its share of the decryption is valid. By lemma 2, these committed proofs can be executed concurrently. Such a method can tolerate up to $n/3$ corruptions.

7 Erasure-free threshold Cramer-Shoup cryptosystem

We exemplify our erasure-free threshold cryptography techniques with a threshold protocol for the Cramer-Shoup cryptosystem (Fig.10). We assume that the key generation was done similarly to Sec. 6, except that all sharings are of the type $\mathsf{RVSS\text{-}data}_{t,g,h}[a,b,c,d,e]$. Since this protocol essentially exponentiates elements v^{-1}, x, y to values that are held with additive sharing, the security of this protocol can be shown in an argument similar to Lemma 9. For full analysis, as well as the key generation protocol, see Jarecki and Lysyanskaya [JL00].

This protocol uses an "additive multiplication" subprotocol MULT (Fig.9), which creates $\mathsf{ADD\text{-}data}$[c], an *additive sharing with polynomial backups* of value $c = ab \bmod q$ from sharings $\mathsf{RVSS\text{-}data}$[a] and $\mathsf{RVSS\text{-}data}$[b]. Note that if α_i's and β_i's are shares of t-degree polynomials f_a, f_b s.t. $f_a(0) = a$, $f_b(0) = b$ then $c = \sum_{i=1}^{n} v_i$ where $v_i = \lambda_i \alpha_i \beta_i \bmod q$ for some interpolation coefficients λ_i (assuming, for simplicity, that $n = 2t+1$). Therefore, the players already hold additive shares v_i of c, but they are not independently distributed. Protocol MULT essentially re-randomizes this additive sharing, as the "2sum-to-2sum" protocol of [FMY99a-b] (except that here all players participate). In the future use of the newly created additive shares c_i of c, the polynomial sharings $\mathsf{RVSS\text{-}data}$[a,b] can serve as backups: If any player P_j misuses or witholds its c_j in the future, these shares are used to reconstruct $v_j = \lambda_j a_j b_j$, and the values c_i's of all other players are adjusted so that $\sum_{i \neq j} c_i = c$.

Input: Sharings RVSS-data[a], RVSS-data[b], values p, q, g, h
Goal: Additive sharing ADD-data[c] of $c = ab = \sum_{i=1}^{n} \lambda_i \alpha_i \beta_i \bmod q$

1. Each player P_i computes its additive share $v_i = \lambda_i \alpha_i \beta_i$ of c, picks $\hat{v}_i \in Z_q$, broadcasts value $F_{v_i}(0) = g^{v_i} h^{\hat{v}_i}$, and proves that v_i in V_i is the product of α_i and $\lambda_i \beta_i$ committed to in $F_a(i)$ and $(F_b(i))^{\lambda_i}$. This is done using a simultaneous proof of Sec.(5.3) with a 3-move public-coin zero-knowledge proof of [CD98].
2. Players perform the "2sum-to-2sum" protocol of [FMY99a-b] for additive re-sharing of shares $v_1, .., v_n$. At the end each P_i computes its new additive share $c_i, \hat{c}_i$ of c, and there are public verification values $F_{c_i}(0) = g^{c_i} h^{\hat{c}_i}$.

Fig. 9. Multiplication MULT : $(\text{RVSS-data}[a], \text{RVSS-data}[b]) \rightarrow \text{ADD-data}[ab]$

Input: Ciphertext x, y, w, σ, v, public key g_1, g_2, C, D, W, values p, q, g, h
Sharings RVSS-data[a,b,c,d,e,s] (i.e. RVSS-data[a],RVSS-data[b],etc.)
Goal: Decrypt cleartext $m = w(v^{-1})^s x^{s(a+c\sigma)-e} y^{s(b+d\sigma)} \bmod p$

1. Each player locally obtains its part of RVSS-data[$a+c\sigma$] and RVSS-data[$b+d\sigma$] from RVSS-data[a,b,c,d] and σ.
2. Let $r = s(a + c\sigma)$ and $z = s(b + d\sigma)$. Players perform two parallel MULT instances to get ADD-data[r] and ADD-data[z] from RVSS-data[s,$a+c\sigma$,$b+d\sigma$]
3. Each P_i broadcasts $m_i = w(v^{-1})^{s_i} x^{r_i - e_i} y^{z_i}$ and proves, using simultaneous proof of Sec.(5.3) with a 3-move public-coin zero-knowledge proof of equality of representation of m_i/w, $F_{s_i}(0)$, $F_{r_i}(0)/F_{e_i}(0)$, and $F_{z_i}(0)$ in appropriate bases made of elements $1, g, h, v^{-1}, x, y$ in G_q.
4. If any player fails, their secret inputs are reconstructed.

Fig. 10. Adaptive Erasure-Free Cramer-Shoup Protocol

ACKNOWLEDGEMENTS

Anna Lysyanskaya is immensely grateful to Victor Shoup who guided much of this research during her stay at IBM Zürich Research lab. She is also very grateful to Christian Cachin for helpful thoughts and discussions. Both authors acknowledge the help and encouragement of Ron Rivest, Shafi Goldwasser, Silvio Micali, Rosario Gennaro, Hugo Krawczyk, Tal Rabin, Klaus Kursawe and Leonid Reyzin. We also thank the anonymous referees for their numerous helpful comments.

In addition, Stanisław Jarecki acknowledges an NTT research grant, and Anna Lysyanskaya acknowledges the support of an NSF graduate fellowship and of the Lucent Technologies GRPW program.

References

[Bea97] Donald Beaver. Plug and play encryption. In *Advances in Cryptology—CRYPTO 97*. Springer-Verlag, 1997.

[BG92] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *Advances in Cryptology—CRYPTO 92*. Springer-Verlag, 1992.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Widgerson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *ACM Symposium on Theory of Computing*, pages 1–10, 1988.

[BH92] D. Beaver and S. Haber. Cryptographic protocols provably secure againts dynamic adversaries. In *Advances in Cryptology—EUROCRYPT 92*, 1992.

[BR97] Mihir Bellare and Phillip Rogaway. Collision-resistant hashing: towards making uowhfs practical. In *Advances in Cryptology—CRYPTO 97*, 1997.

[Bra99] Stefan Brands. Rethinking public-key infrastructures and digital certificates–building in privacy. *Ph.D. dissertation, Technical University of Eindhoven*, 1999.

[Cam98] Jan Camenisch. Group signature schemes and payment systems based on the discrete logarithm problem. *ETH Series in Information Security and Cryptography, vol.2*, 1998.

[Can98] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Theory of Cryptography Library, http://philby.ucsd.edu/cryptolib/1998.html*, 1998.

[CCD88] David Chaum, Claude Crepeau, and Ivan Damgård. Multiparty unconditionally secure protocols. In *Proc. 20th ACM Symp. on Theory of Computing*, pages 11–19, 1988.

[CD98] Ronald Cramer and Ivan Damgård. Zero-knowledge proof for finite field arithmetics, or: Can zero-knowledge be for free. In *Advances in Cryptology—CRYPTO 98*, pages 424–441. Springer-Verlag, 1998.

[CDD+99] Ronald Cramer, Ivan Damgård, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary. In *Advances in Cryptology—EUROCRYPT 99*, 1999.

[CFGN96] Ran Canetti, Uri Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 639–648, 1996.

[CFIJ99] Giovanni Di Crescenzo, Niels Ferguson, Russell Impagliazzo, and Markus Jakobsson. How to forget a secret. In *Proceedings of STACS'99*, 1999.

[CG99] Ran Canetti and Shafi Goldwasser. An efficient threshold public key cryptosystem secure against adaptice chosen ciphertext attack. In *Advances in Cryptology—EUROCRYPT 99*, pages 90–106. Springer-Verlag, 1999.

[CGJ+99a] Ran Canetti, Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Adaptive security for threshold cryptosystems. In *Advances in Cryptology—CRYPTO 99*. Springer-Verlag, 1999.

[CGJ+99b] Ran Canetti, Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Adaptive security for threshold cryptosystems. *http://theory.lcs.mit.edu/~cis/cis-publications.html*, 1999.

[CS98] Ronald Cramer and Victor Shoup. A practical public-key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology—CRYPTO 98*. Springer-Verlag, 1998.

[Des87] Yvo Desmedt. Society and group oriented cryptography. In *Advances in Cryptology—CRYPTO 87*. Springer-Verlag, 1987.

[DF89] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In *Advances in Cryptology—CRYPTO 89*, pages 307–315. Springer-Verlag, 1989.

[DNRS99] Cynthia Dwork, Moni Naor, Omer Reingold, and Larry Stockmeyer. Magic functions. In *40th IEEE Symp. on Foundations of Comp. Science*, 1999.

[FMY99a] Yair Frankel, Philip MacKenzie, and Moti Yung. Adaptively-secure distributed threshold public key systems. In *Proceedings of ESA 99*, 1999.

[FMY99b] Yair Frankel, Philip MacKenzie, and Moti Yung. Adaptively-secure optimal-resilience proactive RSA. In *Advances in Cryptology—ASIACRYPT 99*. Springer-Verlag, 1999.

[GJKR99] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete–log based cryptosystems. In *Advances in Cryptology—EUROCRYPT 99*, pages 295–310, 1999.

[Gol95] Oded Goldreich. Foundations of cryptography: Fragments of a book. *http://theory.lcs.mit.edu/˜oded*, 1995.

[HJJ+97] Amir Herzberg, Markus Jakobsson, Stanisław Jarecki, Hugo Krawczyk, and Moti Yung. Proactive public key and signature systems. In *4th ACM Conf. on Comp. and Comm. Security*, pages 100–110, 1997.

[JL00] Stanislaw Jarecki and Anna Lysyanskaya. Adaptively secure threshold cryptography without erasures. *Theory of Cryptography Library*, 2000.

[Lys00] Anna Lysyanskaya. Threshold cryptography secure against the adaptive adversary, concurrently. *Theory of Cryptography Library*, 2000.

[Ped91] T. Pedersen. A threshold cryptosystem without a trusted party. In *Advances in Cryptology—EUROCRYPT 91*, pages 522–526, 1991.

[Sch91] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[SG98] Victor Shoup and Rosario Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In *Advances in Cryptology—EUROCRYPT 98*. Springer-Verlag, 1998.

[Sho99a] Victor Shoup. A composition theorem for universal one-way hash functions. *IBM Research Report RZ3147*, 1999.

[Sho99b] Victor Shoup. Why chosen ciphertext security matters. *IBM Research Report RZ3076*, 1999.