Universally Composable DKG with Linear Number of Exponentiations

Douglas Wikström

Royal Institute of Technology (KTH),
KTH, Nada, S-100 44 Stockholm, Sweden

Abstract. Until now no distributed discrete-logarithm key generation (DKG) protocol is known to be universally composable. We extend Feldman's verifiable secret sharing scheme to construct such a protocol. Our result holds for static adversaries corrupting a minority of the parties under the Decision Diffie-Hellman assumption in a weak common random string model in which the simulator *does not choose* the common random string.

Our protocol is optimistic. If all parties behave honestly, each party computes $O(3.5k)$ exponentiations, and otherwise each party computes $O(k^2)$ exponentiations, where k is the number of parties. In previous constructions each party always computes $\Omega(k^2)$ exponentiations.

1 Introduction

The ability of a group of parties to jointly generate a public key for which the secret key is shared is a cornerstone of threshold cryptography. Without a method to do this securely the parties must resort to a preliminary phase in which a trusted key generating party is present. In some applications no natural trusted party exists, e.g. electronic voting. When a discrete-logarithm based cryptographic primitive is used, distributed key generation often amounts to generating a public key $y = g^x$ for which the corresponding secret key x is secretly and verifiably shared among the parties. Following Gennaro et al. [14] we call a protocol that does this securely a DKG protocol.

1.1 Previous Work

The problem of constructing a DKG protocol was first investigated by Pedersen [25]. His basic building block was a new non-interactive verifiable secret sharing scheme [24] based on ideas of Feldman [8]. Pedersen DKG has been used as a subprotocol in numerous constructions in the literature, but it has never been verified that Pedersen DKG composes correctly in general. Indeed, Gennaro et al. [14] pointed out that the Pedersen DKG may generate a public key which is biased by the adversary. They also gave a new modified protocol and gave a more careful analysis. Adaptively secure protocols for DKG where given by Canetti et al. [6] and Jarecki and Lysyanskaya [21]. Independently, Frankel, MacKenzie and

C. Blundo and S. Cimato (Eds.): SCN 2004, LNCS 3352, pp. 263–277, 2005.

Yung gave key generation protocols secure against adaptive adversaries in several papers [9, 10, 11, 12]. They also considered threshold variants of RSA. Recently, Gennaro et al. [13] investigated the security of the original Pedersen DKG, i.e. how the adversary can benefit from biasing the public key. They show that under certain circumstances the adversary gains very little from this additional power.

Canetti [5] and independently Pfitzmann and Waidner [26], proposed security frameworks for reactive processes. We use the former framework, i.e. the Universally Composable security framework (UC-security). Both frameworks have composition theorems, and are based on older definitional work. The initial ideal-model based definitional approach for secure function evaluation is informally proposed by Goldreich, Micali, and Wigderson in [17]. The first formalizations appear in Goldwasser and Levin [18], Micali and Rogaway [23], and Beaver [3]. Canetti [4] presents the first definition of security that is preserved under composition. See [4, 5] for an excellent background on these definitions.

1.2 Contribution

We give a protocol that securely realizes the ideal DKG functionality in a universally composable way under the Decision Diffie-Hellman assumption. Thus, our protocol can be plugged as a subprotocol in any setting where a DKG protocol is needed. Our result holds in a very weak common random string model in that the simulator does not choose the common random string.

Let k be the number of parties. Our protocol is optimistic and each party computes only $O(3.5k)$ exponentiations if all parties behave honestly and $O(k^2)$ otherwise. In previous constructions each party computes $\Omega(k^2)$ exponentiations.

Pedersen commitments [25] are not used at any point in our protocol. We think it is particularly nice to see that Feldman's original ideas can be used directly.

We note that in work independent of ours, Abe and Fehr [1] have announced an adaptively UC-secure DKG protocol. Each party computes $\Omega(k^2)$ exponentiations in their protocol, and the security rests on the DDH-assumption.

1.3 Universally Composable Security

Throughout this paper we employ the universally composable security framework (UC-framework) of Canetti [5]. Our result does not depend on technicalities of any particular flavor, but to avoid any ambiguity we review in Appendix A the precise definitions we use. The idea of UC-security is to define security such that if a protocol π "securely realizes" a functionality $\mathcal{F}$, then π can be plugged in as a subprotocol in *any* setting where a functionality $\mathcal{F}$ is needed. Thus, the model allows modular analysis of the security of protocols, and guarantees that the security properties of a protocol is preserved regardless of where it is used.

The framework is formalized by defining a real model in which the protocol π executes, and an ideal model which essentially contains the functionality $\mathcal{F}$ the protocol should realize. The protocol π is said to *securely realize* $\mathcal{F}$ if for each adversary $\mathcal{A}$ in the real model there exists an ideal adversary $\mathcal{S}$ in the ideal model such that no environment $\mathcal{Z}$ can distinguish between executions in the

real and ideal models. A hybrid model is a real model where the protocol π is given access to additional functionalities $\mathcal{F}'$. The protocol π'', where each call to $\mathcal{F}'$ is replaced by an invokation of a protocol π' is called the *composition of* π *and* π'. The UC-composition theorem says that π'' securely realizes $\mathcal{F}$ if π securely realizes $\mathcal{F}$ in the $\mathcal{F}'$-hybrid model and π' securely realizes $\mathcal{F}'$.

1.4 Notation

Throughout, $M_1, \ldots, M_k$ denote the participating parties, which are modeled as interactive Turing machines. We abuse notation and use M_j to denote both the machines themselves and their identity. We write $\mathcal{M}_{k/2}$ to denote the set of *static* polynomial time *non-uniform* Turing machines that can corrupt a minority of the parties.

We use the term "randomly" instead of "uniformly and independently at random". We assume that G_q is a group of prime order q with generator g for which the Decision Diffie-Hellman Assumption holds, e.g. a subgroup G_q of prime order q of $\mathbb{Z}_p^*$ for some $p = \kappa q + 1$. We take $\log p = n$ to be our security parameter, and assume that computing an exponentiation in G_q takes time corresponding to computing at least n multiplications in G_q. This allows us to express the complexity of our procotol in terms of the number of exponentiations computed.

Assumption 1 (Decision Diffie-Hellman). *Let $e_1, e_2, e_3 \in \mathbb{Z}_q$ be randomly chosen. The (non-uniform) Decision Diffie-Hellman assumption for G_q states that for all polynomial time non-uniform Turing machines A, $\forall c > 0$, $\exists n_0$, such that for $n > n_0$:*

$$|\Pr[A(g^{e_1}, g^{e_2}, g^{e_3}) = 1] - \Pr[A(g^{e_1}, g^{e_2}, g^{e_1 e_2}) = 1]| < \tfrac{1}{n^c}.$$

We use $\mathcal{C}_\mathcal{I}$ to denote the ideal communication model. It routes authenticated messages between the parties, the ideal adversary, and the functionalities. The first component of a list handed to $\mathcal{C}_\mathcal{I}$ is the identity of the receiver. The adversary decides when $\mathcal{C}_\mathcal{I}$ delivers messages. The notion of a bulletin board is intuitively clear.

Functionality 1 (Bulletin Board (cf. [28])). The ideal *bulletin board* functionality, $\mathcal{F}_{\text{BB}}$, running with parties $M_1, \ldots, M_k$ and ideal adversary $\mathcal{S}$.

1. $\mathcal{F}_{\text{BB}}$ holds a database indexed on integers. Initialize a counter $c = 0$.
2. On receiving $(M_i, \texttt{Write}, m_i)$, $m_i \in \{0,1\}^*$, from $\mathcal{C}_\mathcal{I}$, store (M_i, m_i) under c in the database, hand $(\mathcal{S}, \texttt{Write}, c, M_i, m_i)$ to $\mathcal{C}_\mathcal{I}$, and set $c \leftarrow c + 1$.
3. Upon receiving $(M_j, \texttt{Read}, c)$ from $\mathcal{C}_\mathcal{I}$ check if a tuple (M_i, m_i) is stored in the database under c. If so hand $((\mathcal{S}, M_j, \texttt{Read}, c, M_i, m), (M_j, \texttt{Read}, c, M_i, m_i))$ to $\mathcal{C}_\mathcal{I}$. If not, hand $((\mathcal{S}, M_j, \texttt{NoRead}, c), (M_j, \texttt{NoRead}, c))$ to $\mathcal{C}_\mathcal{I}$.

Goldwasser and Lindell [19] show that authenticated broadcast can be securely realized with respect to *blocking* $\mathcal{M}_{k/2}$-adversaries. On the other hand Lindell, Lysyanskaya and Rabin [22] show that composable authenticated broadcast can not be realized for *non-blocking* $\mathcal{M}_B$-adversaries if $B > k/3$. A non-blocking adversary is an adversary that never delays the delivery of messages to honest parties indefinitely. The following lemma follows straightforwardly from [19].

Lemma 1. *There exists a protocol π_{BB} that securely realizes $\mathcal{F}_{\mathrm{BB}}$ with respect to* blocking *$\mathcal{M}_{k/2}$-adversaries.*

In many constructions the parties are assumed to be able to communicate secretly with each other. In the UC-framework this was modeled by Canetti [5]. Below we give a slightly modified variant, better suited for our setting.

Functionality 2 (Multiple Message Transmission). The ideal multiple message transmission, $\mathcal{F}_{\mathrm{MMT}}$, with parties $M_1, \ldots, M_k$ and ideal adversary $\mathcal{S}$.

1. In the first activation expect to receive a value (`Receiver`) from some party M_j. Then hand $((\mathcal{S}, \texttt{Receiver}, M_j), \{(M_i, \texttt{Receiver}, M_j)\}_{i=1}^{k})$ to $\mathcal{C}_{\mathcal{I}}$.
2. Upon receiving $(M_j, \texttt{Send}, M_i, m_j)$ from $\mathcal{C}_{\mathcal{I}}$, hand $((\mathcal{S}, M_j, \texttt{Send}, M_i, |m_j|), (M_i, M_j, m_j))$ to $\mathcal{C}_{\mathcal{I}}$.

From Claim 16 in Canetti [5] and the fact that the Cramer-Shoup cryptosystem [7] is chosen ciphertext secure in the sense of Rackoff and Simon [27] under the Decision Diffie-Hellman assumption in G_q, the lemma below follows.

Lemma 2. *There exists a protocol π_{MMT} that securely realizes $\mathcal{F}_{\mathrm{MMT}}$ under the Decision Diffie-Hellman assumption in G_q.*

A common assumption used in the construction of protocols is the existence of a common random string (CRS). A common reference string is different from a CRS in that it may have additional structure, or be generated together with a trapdoor which allows easy simulation. Previous DKG-protocols require the existence of a common reference string $g, h \in G_q$ such that the simulator knows $\log_g h$. When this is not the case the protocols can not be simulated. We make no such assumptions. Our simulator is *not* allowed to choose the CRS. Thus, our CRS can truly be a random string defined by a physical experiment.

Functionality 3 (Common Random String (CRS)). The ideal common random string, $\mathcal{F}_{\mathrm{CRS}}$, running with parties $M_1, \ldots, M_k$ and ideal adversary $\mathcal{S}$ simply chooses $h_1, h_2, h_3 \in G_q$ randomly and hands $((\mathcal{S}, \texttt{CRS}, h_1, h_2, h_3), \{(M_j, \texttt{CRS}, h_1, h_2, h_3)\}_{j=1}^{k})$ to $\mathcal{C}_{\mathcal{I}}$.

2 Distributed Key Generation

The functionality below captures the notion defined by Gennaro et al. [14], but in the language of the UC-framework. A public key $y = g^x$ is generated and given to all parties. Each party also receives a share s_j of the secret key x.

Functionality 4 (Distributed Key Generation (DKG)). The ideal *Distributed Key Generation over G_q*, $\mathcal{F}_{\mathrm{DKG}}$, running with generators $M_1, \ldots, M_k$, and ideal adversary $\mathcal{S}$ proceeds as follows. Let $t = \lceil k/2 - 1 \rceil$.

1. Wait for $(\texttt{CorruptShares}, \{j, s_j\}_{j \in I_M})$ from $\mathcal{S}$, where I_M is the set of indices of corrupted parties.

2. Choose $a_\iota \in \mathbb{Z}_q$ randomly under the restriction $a(j) = s_j$ for $j \in I_M$, where $a(z) = \sum_{\iota=0}^{t} a_\iota z^\iota$. Then define $s_j = a(j)$ for $j \notin I_M$, and set $y = g^{a_0}$.
3. Hand $((\mathcal{S}, \texttt{PublicKey}, y), \{(\texttt{PublicKey}, y, s_j)\}_{j=1}^{k})$ to $\mathcal{C}_\mathcal{I}$.

Note that the adversary chooses the shares handed to corrupted parties.

Recall the verifiable secret sharing scheme of Feldman [8]. A dealer shares a secret $a_0 \in \mathbb{Z}_q$ by choosing $a_\iota \in \mathbb{Z}_q$ randomly and forming a polynomial $a(z) = \sum_{\iota=0}^{t} a_\iota z^\iota$ of degree t. Then it publishes $\alpha_\iota = g^{a_\iota}$ and hands a share $s_j = a(j)$ to the j:th party. This allows the receiver of a share s_j to verify its correctness by checking that $g^{s_j} = \prod_{\iota=0}^{t} \alpha_\iota^{j^\iota}$. If a share is not correct the receiver complains and forces the dealer to publish a correct share. To recover the secret the receivers simply publish their shares. This allows anybody to find a set of correct shares and Lagrange interpolate the secret. The distribution step of Feldman's protocol gives a way to share the secret key x corresponding to a public key $\alpha_0 = g^x$.

Next we give an informal description of our protocol. The parties are partitioned into three sets and each set is assigned a random generator $h_f \in G_q$. Each party runs a copy of Feldman's protocol, but using its set's generator. Instead of verifying each individual dealer's shares and public information, they are combined within each set of parties. This allows efficient verification. The basic idea behind this trick was taken from Gennaro et al. [15]. If some party is malicious, the efficient way of verifying shares is abandoned and individual verifications are performed. From this each party M_j computes a combined share s_j, the sum of all correct shares it received. Then each party publishes $\beta_j = g^{s_j}$. Note that this time all parties use the common generator g. Then the parties verify the correctness of the β_j's and construct a joint key $y = g^x$, for which x is the secret to which the combined shares s_j correspond. If the verification fails each party is essentially required to prove that its β_j is correct. This allows all parties to agree on a set of correct β_j from which the joint key can be constructed.

Let $\{\Omega_1, \Omega_2, \Omega_3\}$ be a partition of $\{1, \ldots, k\}$ such that $||\Omega_f| - |\Omega_{f'}|| \leq 1$ for $f \neq f'$. Define $f(j)$ to be the value of f such that $j \in \Omega_f$. All parties always verify that their input is contained in G_q or $\mathbb{Z}_q$ as expected by the protocol.

Protocol 1 (Distributed Key Generation (DKG)). Let $t = \lceil k/2 - 1 \rceil$. The Distributed Key Generation protocol with generators $M_1, \ldots, M_k$.

Preliminary Phase

1. Hand $(\texttt{Receiver})$ to $\mathcal{F}_{\text{MMT}}$.
2. Wait for $(\texttt{Receiver}, M_l)$ for $l = 1, \ldots, k$ from $\mathcal{F}_{\text{MMT}}$.
3. Wait for $(\texttt{CRS}, h_1, h_2, h_3)$ from $\mathcal{F}_{\text{CRS}}$.

Key Generation Phase

4. Define f by $j \in \Omega_f$. Choose $a_{j,\iota} \in \mathbb{Z}_q$ randomly, and define $a_j(z) = \sum_{\iota=0}^{t} a_{j,\iota} z^\iota$, $\alpha_{j,\iota} = h_f^{a_{j,\iota}}$, and $s_{j,l} = a_j(l)$. Hand $(\texttt{Send}, M_l, \texttt{Share}, s_{j,l})$ to $\mathcal{F}_{\text{MMT}}$ for $l \neq j$ and $(\texttt{Write}, \texttt{PublicElements}, \{\alpha_{j,\iota}\}_{\iota=0}^{t})$ to $\mathcal{F}_{\text{BB}}$.
5. Wait until $(M_l, \texttt{PublicElements}, \{\alpha_{l,\iota}\}_{\iota=0}^{t})$ appears on $\mathcal{F}_{\text{BB}}$ and a message $(M_l, \texttt{Share}, s_{l,j})$ is received from $\mathcal{F}_{\text{MMT}}$ for $l \neq j$. Choose a random n-subset

$A_j \subset \{0,\ldots,t\}$ (if $t < n$ then $A_j = [k]$), compute $\theta_{j,f,\iota} = \prod_{l\in\Omega_f} \alpha_{l,\iota}$ for $f = 1,2,3$ and $\iota \in A_j$. Then hand $(\texttt{Write}, \texttt{Products}, \{\theta_{j,f,\iota}\}_{\iota\in A_j, f\in\{1,2,3\}})$ to $\mathcal{F}_{\text{BB}}$.

6. Wait until $(M_l, \texttt{Products}, \{\theta_{l,f,\iota}\}_{\iota\in A_l, f\in\{1,2,3\}})$ appears on $\mathcal{F}_{\text{BB}}$ for $l \neq j$. Then verify that $\theta_{l,f,\iota} = \theta_{l',f,\iota}$ for $\iota \in A_l \cap A_{l'}$. If not go to Step 12. If so define $\theta_{f,\iota} = \theta_{l,f,\iota}$ for any l such that $\iota \in A_l$.
7. Verify that $h_f^{\sum_{l\in\Omega_f} s_{l,j}} = \prod_{\iota=0}^{t} \theta_{f,\iota}^{j^\iota}$, for $f = 1,2,3$. If so, hand $(\texttt{Write}, \texttt{Complaints}, \emptyset)$ to $\mathcal{F}_{\text{BB}}$. Otherwise, go to Step 12.
8. Wait until $(M_l, \texttt{Complaints}, \Delta_l)$ appears on $\mathcal{F}_{\text{BB}}$ for $l \neq j$. If all $\Delta_l = \emptyset$ set $I_1 = \{1,\ldots,k\}$. Otherwise, go to Step 13.
9. Define $s_j = \sum_{l\in I_1} s_{l,j}$ and set $\beta_j = g^{s_j}$. Then hand $(\texttt{Write}, \texttt{ConstructPublicKey}, \beta_j)$ to $\mathcal{F}_{\text{BB}}$.
10. Wait until $(M_l, \texttt{ConstructPublicKey}, \beta_l)$ appears on $\mathcal{F}_{\text{BB}}$ for $l \in I_1$. Verify that $\beta_j = \prod_{i=1}^{t+1} \beta_i^{\prod_{l\neq i} \frac{l-j}{l-i}}$. If so set $I_2 = \{1,\ldots,t+1\}$. If not, go to Step 15.
11. Define $y = \prod_{i\in I_2} \beta_i^{\prod_{l\neq i} \frac{l}{l-i}}$ and output $(\texttt{PublicKey}, y, s_j)$.

Handle Cheating with the $s_{l,i}$ and $\alpha_{l,\iota}$.

12. Verify for $l = 1,\ldots,k$ that $h_{f(l)}^{s_{l,j}} = \prod_{\iota=0}^{t} \alpha_{l,\iota}^{j^\iota}$. Let Δ_j be the set of indices l for which inequality holds. Then hand $(\texttt{Write}, \texttt{Complaints}, \Delta_j)$ to $\mathcal{F}_{\text{BB}}$.
13. Wait until $(M_l, \texttt{Complaints}, \Delta_l)$ appears on $\mathcal{F}_{\text{BB}}$ for $l \neq j$. Let $\Gamma_j = \{l \mid j \in \Delta_l\}$. Then hand $(\texttt{Write}, \texttt{Refutes}, \{s_{j,l}\}_{l\in\Gamma_j})$ to $\mathcal{F}_{\text{BB}}$.
14. Wait until $(M_l, \texttt{Refutes}, \{s_{l,i}\}_{i\in\Gamma_l})$ appears on $\mathcal{F}_{\text{BB}}$ for $l \neq j$ and replace old values of $s_{l,j}$ with the new. Let I_1 be the set of l such that $h_{f(l)}^{s_{l,i}} = \prod_{\iota=0}^{t} \alpha_{l,\iota}^{i^\iota}$ for all i such that $(l,i) \in [k] \times \{j\} \cup \bigcup_{i'=1}^{k} (\Delta_{i'} \times \{i'\})$. Then go to Step 9.

Handle Cheating with the β_l.

15. Set $c_{f,j,0} = \sum_{l\in\Omega_f\cap I_1} s_{l,j}$ and choose $c_{f,j,\iota} \in \mathbb{Z}_q$ for $\iota > 0$ randomly. Define $c_{f,j}(z) = \sum_{\iota=0}^{t} c_{f,j,\iota} z^\iota$, $\gamma_{j,\iota} = g^{c_{1,j,\iota}+c_{2,j,\iota}+c_{3,j,\iota}}$, $\delta_{f,j,\iota} = h_f^{c_{f,j,\iota}}$, and $\zeta_{f,j,l} = c_{f,j}(l)$. Then hand $(\texttt{Send}, M_l, \texttt{Share2}, (\zeta_{1,j,l}, \zeta_{2,j,l}, \zeta_{3,j,l}))$ to $\mathcal{F}_{\text{MMT}}$ for $l \in I_1$ and hand $(\texttt{Write}, \texttt{PublicElements2}, \{\gamma_{j,\iota}, \delta_{1,j,\iota}, \delta_{2,j,\iota}, \delta_{3,j,\iota}\}_{\iota=1}^{t})$ to $\mathcal{F}_{\text{BB}}$.
16. Wait for $(M_l, \texttt{PublicElements2}, \{\gamma_{l,\iota}, \delta_{1,j,\iota}, \delta_{2,j,\iota}, \delta_{3,j,\iota}\}_{\iota=1}^{t})$ on $\mathcal{F}_{\text{BB}}$ for $l \in I_1$. Set $\gamma_{l,0} = \beta_l$ and $\delta_{f,l,0} = \prod_{i\in\Omega_f\cap I_1} \alpha_{i,\iota}$. Verify $g^{\zeta_{1,l,j}+\zeta_{2,l,j}+\zeta_{3,l,j}} = \prod_{\iota=0}^{t} \gamma_{l,\iota}^{j^\iota}$ and $h_f^{\zeta_{f,l,j}} = \prod_{\iota=0}^{t} \delta_{f,l,\iota}^{j^\iota}$. Let Δ'_j be the set of indices for which the verification fails. Then hand $(\texttt{Write}, \texttt{Complaints2}, \Delta'_j)$ to $\mathcal{F}_{\text{BB}}$.
17. Wait until $(M_l, \texttt{Complaints2}, \Delta'_l)$ appears on $\mathcal{F}_{\text{BB}}$ for $l \in I_1$. Let $\Gamma'_j = \{l \mid j \in \Delta'_l\}$. Then hand $(\texttt{Write}, \texttt{Refutes2}, \{\zeta_{1,j,l}, \zeta_{2,j,l}, \zeta_{3,j,l}\}_{l\in\Gamma'_j})$ to $\mathcal{F}_{\text{BB}}$.
18. Wait until $(M_l, \texttt{Refutes2}, \{\zeta_{1,j,l}, \zeta_{2,j,l}, \zeta_{3,j,l}\}_{i\in\Gamma_l})$ appears on $\mathcal{F}_{\text{BB}}$ for $l \in I_1$ and replace the old values of $\zeta_{f,l,j}$ with the new. Let I_2 be the lexicographically first set of l such that $g^{\zeta_{1,l,i}+\zeta_{2,l,i}+\zeta_{3,l,i}} = \prod_{\iota=0}^{t} \gamma_{l,\iota}^{i^\iota}$ and $h^{\zeta_{f,l,i}} = \prod_{\iota=0}^{t} \delta_{f,l,\iota}^{i^\iota}$ for all i such that $(l,i) \in I_1 \times \{j\} \cup \bigcup_{i'=1}^{k} (\Delta'_{i'} \times \{i'\})$. Then go to Step 11.

All shares corresponding to a partition Ω_f are verified together. One can also consider verifying smaller sets together if some cheating is expected. Then if

cheating is detected, fewer shares have to be verified individually. This changes the security analysis only slightly.

Remark 1. Theoretically, it suffices that each party chooses an $e(n)$-subset in Step 5, where $e(n)$ is a function such that $e(n)/\log n = \Omega(1)$.

Theorem 1. *The protocol π above securely realizes $\mathcal{F}_{\mathrm{DKG}}$ with respect to $\mathcal{M}_{k/2}$-adversaries in the $(\mathcal{F}_{\mathrm{BB}}, \mathcal{F}_{\mathrm{MMT}}, \mathcal{F}_{\mathrm{CRS}})$-hybrid model under the DDH-assumption.*

If all parties behave honestly, each party computes $O(3.5k)$ exponentiations in G_q. In the worst case each party computes $O(k^2)$ exponentiations.

Corollary 1. *If π is composed with π_{MMT}, the result holds in the $(\mathcal{F}_{\mathrm{BB}}, \mathcal{F}_{\mathrm{CRS}})$-hybrid model. If also composed with π_{BB} the result holds in the $\mathcal{F}_{\mathrm{CRS}}$-hybrid model for* blocking *adversaries.*

Some intuition behind the construction and for the proof of Theorem 1 follows. Since the adversary could potentially generate the shares it distributes *after* receiving shares from all honest parties, it can choose the shares it distributes such that the combined shares s_j of corrupted parties take on certain values. This is why we must allow the ideal adversary to do this in the DKG functionality as well (the distribution of y is unbiased).

Already after the public elements $\alpha_{j,\iota}$ are published the simulator can extract the secrets of corrupt parties, compute the resulting final combined shares, and feed them to the ideal functionality. To simulate honest dummy parties, the ideal adversary generates in the second step public elements β_j that appear correct to the corrupt parties, but they are carefully chosen such that the public key y output by a protocol execution is identical to the public key output by the ideal functionality. This implies that the $\alpha_{j,\iota}$ and β_j are inconsistent. This fact must be hidden from the adversary and environment such that the latter can not distinguish a simulation from a real execution. This must hold despite that the environment knows all shares s_j at the end of an execution.

The use of three independent generators h_1, h_2 and h_3 in the first phase ensures that no adversary or environment can check if the final shares s_j correspond to the public elements $\alpha_{l,\iota}$. The adversary is essentially given a tuple $(h_1, h_2, h_1^{e_1}, h_2^{e_2}, (1-b)e_3 + b(e_1+e_2))$, where $h_1, h_2 \in G_q$ and $e_1, e_2, e_3 \in \mathbb{Z}_q$ are random, and must guess b. This problem is related to the DDH-problem as follows. Consider a tuple (h_1, u, v, w), where $(u, v, w) = (h_1^{e_1}, h_1^{e_2}, h_1^{be_1e_2+(1-b)e_3})$. This is a DDH-tuple if $b = 1$. Define $h_2 = u = h_1^{e_1}$, $U = h_1^{\sigma}/v = h_1^{\sigma-e_2}$, $V = w = h_2^{(1-b)e_3+be_2}$, and $W = \sigma$ for a random σ. Then (h_1, h_2, U, V, W) is a tuple of the first type and $b = 1$ precisely when (u, v, w) is a DDH-triple. In fact we have translated an instance of the DDH-problem to an instance of a problem that the adversary and environment must solve to distinguish a simulation from a real execution of our protocol.

Although the UC-framework allows it, we have not used any trapdoor for the common random string in the simulation. In fact the common random string is not even chosen by the simulator. Thus, we feel that our use of the common random string is very mild.

Proof (Theorem 1 and Corollary 1). It is easy to verify the complexity claim in the non-optimistic case by counting.

The optimistic case is not as obvious. In Step 4 we invoke a corollary in Section 8.5 in Aho, Hopcroft, and Ullman [2] which says that kth degree polynomial over $\mathbb{Z}_q$ can be evaluated in k points using at most $k \log^2 k$ arithmetic operations (i.e. $+$, $-$, $\times$, $/$) in $\mathbb{Z}_q$. Thus, this step is performed in time corresponding to compute k exponentiations.

In Step 5 each party computes n products, each having k factors. This corresponds to computing k exponentiations.

In Step 7 the exponents j^ι are computed iteratively using the recursion $j^\iota = j \cdot j^{\iota-1}$. The cost is at most k multiplications, and ignored. Similarly the cost of the multiplication of the exponentiated elements is ignored. Thus, the cost of this step is $k/2$ exponentiations, since this is how many ordinary exponentiations are performed.

In Step 10 and Step 11 k ordinary exponentiations are computed, but we must also consider how to compute the exponents in these products. We note that the products are almost factorial (here it is in fact necessary to have $I_2 = \{1, \ldots, t+1\}$), i.e. $\prod_{l \neq i}(l-i) = (1-i)(2-i)\cdot\ldots\cdot((i-1)-i)((i+1)-i)\cdot\ldots\cdot(t+1-i)$. We compute all factorials $1!, 2!, 3!, \ldots, (t+1)!$ in $\mathbb{Z}_q$ and their inverses. Then each of our exponents can be formed using only 4 multiplications. Thus, all exponents can be computed using at most $O(k)$ multiplications, and the total cost of the two steps corresponds to k exponentiations.

In total each party computes $O(3.5k)$ exponentiations.

We construct an ideal adversary $\mathcal{S}$ that runs any hybrid adversary $\mathcal{A}$ as a blackbox. Then we show that if $\mathcal{S}$ does not imply that the protocol is secure, the DDH-assumption is broken.

The Ideal Adversary $\mathcal{S}$. Let I_M be the set of indices of generators corrupted by $\mathcal{A}$. The ideal adversary $\mathcal{S}$ corrupts the dummy generators $\overline{M}_j$ for $j \in I_M$. The ideal adversary is best described by starting with a copy of the original hybrid ITM-graph $(V, E) = \mathcal{Z}'(\mathcal{H}(\mathcal{A}, \pi^{\overline{\pi}^{(\mathcal{F}_{\mathrm{BB}}, \mathcal{F}_{\mathrm{MMT}}, \mathcal{F}_{\mathrm{CRS}})}}))$ where $\mathcal{Z}$ is replaced by a machine $\mathcal{Z}'$ that we define below. The adversary $\mathcal{S}$ simulates all machines in V except those in $\mathcal{A}'$, and the corrupted machines M_j for $j \in I_M$ under $\mathcal{A}'$:s control.

Simulation of Links $(\mathcal{Z}, \mathcal{A})$, $(\mathcal{Z}, M_j)$ *for* $j \in I_M$. $\mathcal{S}$ simulates $\mathcal{Z}'$ and $\overline{M}_j$ for $j \in I_M$, such that it appears as if $\mathcal{Z}$ and $\mathcal{A}$, and $\mathcal{Z}$ and M_j for $j \in I_M$ are linked directly.

1. When $\mathcal{Z}'$ receives m from $\mathcal{A}$, m is written to $\mathcal{Z}$, by $\mathcal{S}$. When $\mathcal{S}$ receives m from $\mathcal{Z}$, m is written to $\mathcal{A}$ by $\mathcal{Z}'$. This is equivalent to that $\mathcal{Z}$ and $\mathcal{A}$ are linked directly.
2. When $\mathcal{Z}'$ receives m from M_j for $j \in I_M$, m is written to $\mathcal{Z}$ by $\overline{M}_j$. When $\overline{M}_j$, $j \in I_M$, receives m from $\mathcal{Z}$, m is written to M_j by $\mathcal{Z}'$. This is equivalent to that $\mathcal{Z}$ and M_j are linked directly for $j \in I_M$.

Extraction from a Corrupt Generator. Note that since the last t generators M_j which distributes their shares $s_{j,l}$ may be corrupted, the adversary can choose

$s_{j,l}$ for $j \in I_M$ such that $s_j = \sum_{l \in I_1} s_{l,j}$ takes on any values of its choice. The ideal adversary $\mathcal{S}$ must somehow extract the s_j for $j \in I_M$ and then hand them to $\mathcal{F}_{\text{DKG}}$ to ensure that the shares s_j for $j = 1, \ldots, k$ that eventually ends up at the environment are consistent with the output public key y.

When the last $(\texttt{Write}, \texttt{Complaints}, \Delta_j)$ message before executing Step 9 appears on $\mathcal{F}_{\text{BB}}$ for $j = 1, \ldots, k$, interrupt the simulation of $\mathcal{F}_{\text{BB}}$.

There are two cases. If there were no complaints, we claim that $h_f^{\sum_{l \in \Omega_f} s_{l,j}} = \prod_{\iota=0}^{t} (\prod_{l \in \Omega_f} \alpha_{l,\iota})^{j^\iota}$ is satisfied for $j \notin I_M$ and $f = 1, 2, 3$. We argue that $\theta_{f,\iota} = \prod_{l \in \Omega_f} \alpha_{l,\iota}$. This is clearly the case if for each $\iota = 0, \ldots, t$ there exists an honest party M_j such that $\iota \in A_j$. Thus, the probability of failure is bounded by $\Pr[\exists \iota \in [0,t], \forall j \notin I_M : \iota \notin A_j] \leq \sum_{\iota=0}^{t+1} \Pr[\forall j \notin I_M : \iota \notin A_j]$, where we used the union bound. From independence follows that the latter quantity equals $(t+1)\Pr[\iota \notin A_j]^{t+1}$ (for some arbitrary $\iota \in \{1, \ldots, t+1\}$ and $j \notin I_M$). By construction $\Pr[\iota \notin A_j] = (1 - \frac{n}{k-n})$, which implies that the probability of failure is bounded by $(t+1)(1 - \frac{n}{k-n})^{k/2} \leq (t+1)e^{-n/2}$, which is negligible. Thus, with overwhelming probability, $\theta_{f,\iota} = \prod_{l \in \Omega_f} \alpha_{l,\iota}$. This implies that we can Lagrange interpolate $b_{f,i} = \sum_{l \in \Omega_f} (\sum_{j=1}^{t+1} s_{l,j} \prod_{l \neq j} \frac{i-l}{j-l}) = \sum_{l \in \Omega_f} s_{l,i}$ for $f = 1, 2, 3$ and compute $s_i = b_{1,i} + b_{2,i} + b_{3,i}$ for $i \in I_M$. If there were complaints, we have for $l \in I_1$ and $j \notin I_M$ that $h_{f(l)}^{s_{l,j}} = \prod_{\iota=0}^{t} \alpha_{l,\iota}^{j^\iota}$. This implies that the equation above holds for the new values of $s_{l,j}$ and we can Lagrange interpolate $s_i = \sum_{f=1}^{3} \sum_{l \in \Omega_f \cap I_1} s_{l,i}$ similarly to the above. To summarize, $\mathcal{S}$ can always extract s_j for $j \in I_M$ before deciding on β_j values.

The ideal adversary $\mathcal{S}$ hands $(\texttt{CorruptShares}, \{j, s_j\}_{j \in I_M})$ to $\mathcal{F}_{\text{DKG}}$. $\mathcal{F}_{\text{DKG}}$ then returns $(\texttt{PublicKey}, y)$. Below we describe the computations performed by $\mathcal{S}$ before the simulation of $\mathcal{F}_{\text{BB}}$ continues.

Simulation of an Honest Generator. The next problem facing the ideal adversary $\mathcal{S}$ is how to simulate the honest generators M_j for $j \notin I_M$ such that the corrupt generators M_j for $j \in I_M$ output the same public key y as that output by the dummy generators $\overline{M}_j$ for $j \notin I_M$. The latter generators are beyond $\mathcal{S}$'s control and simply forwards the output from $\mathcal{F}_{\text{DKG}}$. Intuitively, $\mathcal{S}$ must "lie" at some point, since it is already committed to a public key by the public $\alpha_{j,\iota}$ elements on $\mathcal{F}_{\text{BB}}$. The "lie" must be carefully constructed such that the adversary can not identify it, and it must be constructed not knowing $\log_g y$.

First it computes $\beta'_j = g^{s_j}$ for $j \in I_M$ and sets $\beta'_0 = y$ and $I'_M = I_M \cup \{0\}$. These are the β'_j for $j \in I_M$ that *should* later be published by the corrupted generators if they behave honestly. Then $\mathcal{S}$ computes $\beta'_j = \prod_{i \in I'_M} (\beta'_i)^{\prod_{l \neq i} \frac{j-l}{i-l}}$, for $j \notin I_M$, and replace β_j with β'_j in the simulation of the honest generators M_j for $j \notin I_M$. The construction ensures that $\beta'_j = \prod_{i=1}^{t+1} (\beta'_i)^{\prod_{l \neq i} \frac{l-j}{l-i}}$. The simulated honest parties M_j for $j \notin I_M$ are instructed not to complain if $\beta_i = \beta'_i$ for $i \notin I_M$.

However, it may be the case that $\beta'_i \neq \beta_i$ for $i \in I_M$, in which case the honest generators must also simulate the handling cheating with the β_i. This is done using the same technique as above. $\mathcal{S}$ chooses $\zeta_{f,j,l}$ randomly for $l \in I_M$

and sets $\zeta_{f,j,l} = 1$ for $j \notin I_M$. Then it replaces the original values of $\gamma_{j,l}$ and $\delta_{f,j,l}$ for $j \notin I_M$ by $\gamma_{j,l} = \prod_{i \in I'_M} \gamma_{j,i}^{\prod_{i' \neq i} \frac{l-i'}{i-i'}}$ and $\delta_{f,j,l} = \prod_{i \in I'_M} \delta_{f,j,i}^{\prod_{i' \neq i} \frac{l-i'}{i-i'}}$. This ensures that $g^{\zeta_{1,l,j}+\zeta_{2,l,j}+\zeta_{3,l,j}} = \prod_{\iota=0}^{t} \gamma_{l,\iota}^{j^\iota}$ and $h_f^{\zeta_{f,l,j}} = \prod_{\iota=0}^{t} \delta_{f,l,\iota}^{j^\iota}$, for $j \in I_M$. The simulated honest generators are instructed to not complain despite that the above equations does not hold for their values of $\zeta_{j,l}$, when $j, l \notin I_M$. This implies that no $\zeta_{f,j,l}$ for any $j \notin I_M$ is ever published. Regardless if β_j for $j \in I_M$ are correct or not we have $y = \prod_{i \in I_2} \beta_i^{\prod_{l \neq i} \frac{l}{l-i}}$ where y is the value output by $\mathcal{F}_{\text{DKG}}$. At this point the simulation of $\mathcal{F}_{\text{BB}}$ is continued.

Reaching a Contradiction. Suppose that $\mathcal{S}$ does not imply the security of the protocol. Then there exists a hybrid adversary $\mathcal{A}' = \mathcal{A}^{(\mathcal{S}_{\text{BB}}, \mathcal{S}_{\text{MMT}}, \mathcal{S}_{\text{CRS}})}$, an environment $\mathcal{Z}$ with auxiliary input $z = \{z_n\}$, a constant $c > 0$ and an infinite index set $\mathcal{N} \subset \mathbb{N}$ such that for $n \in \mathcal{N}$: $|\Pr[\mathcal{Z}_z(\mathcal{I}(\mathcal{S}, \overline{\pi}^{\mathcal{F}_{\text{DKG}}})) = 1] - \Pr[\mathcal{Z}_z(\mathcal{H}(\mathcal{A}', \pi^{(\overline{\pi}_1^{\mathcal{F}_{\text{BB}}}, \overline{\pi}_2^{\mathcal{F}_{\text{MMT}}}, \overline{\pi}_3^{\mathcal{F}_{\text{CRS}}})})) = 1]| \geq \frac{1}{n^c}$, where $\mathcal{S}$ runs $\mathcal{A}'$ as a black-box as described above, i.e. $\mathcal{S} = \mathcal{S}(\mathcal{A}')$.

Defining the Distinguisher. We are now ready to define a distinguisher D that contradicts the DDH assumption. D is confronted with the following test. An oracle first chooses $e_1, e_2, e_3 \in \mathbb{Z}_q$ and a bit $b \in \{0, 1\}$ randomly and defines $(u, v, w) = (h_1^{e_1}, h_1^{e_2}, h_1^{be_1 e_2 + (1-b)e_3})$. Then D is given (u, v, w) and must guess b.

There exists $j \neq i$ such that $i, j \notin I_M$ and $f(i) \neq f(j)$. Without loss we assume that $1, 2 \notin I_M$ and $f(1) = 1$ and $f(2) = 2$.

D does the following. It sets $h_2 = u$, and generates a random h_3. Then it simulates all machines and ideal functionalities as described above, except that

1. The simulation of M_1 and M_2 is special and depends on (u, v, w).
   $\mathcal{S}$ chooses $s_{j,l}$ randomly and defines $\omega_{j,l} = h_j^{s_{j,l}}$ for $j = 1, 2$ and $l \in I_M$. Then it chooses σ randomly in $\mathbb{Z}_q$ and sets $\omega_{1,0} = h_1^\sigma / v$ and $\omega_{2,0} = w$. Then it computes $s_{(1,2),l} = s_{1,l} + s_{2,l}$ for $l \in I_M$ and sets $s_{(1,2),0} = \sigma$. This allows the definition of $s_{(1,2),l} = \sum_{i \in I'_M} s_{(1,2),i} \prod_{j \neq i} \frac{l-i}{j-i}$, for $l \notin I_M$. Set $s_{1,l} = \log_{h_1} \omega_{1,l}$ and $s_{2,l} = \log_{h_2} \omega_{2,l}$ for $l \notin I_M$ (inclusive $l = 0$). These values are not known by $\mathcal{S}$, but we can still consider the equation system $(s_{j,l})_{l \in I'_M} = (l^\iota)_{l \in I'_M, \iota \in [0,t]} (a_{j,\iota})_{\iota \in [0,t]}$. Denote by $(d_{l,\iota})_{l \in I'_M, \iota \in [0,t]}$ the inverse of $(l^\iota)_{l \in I'_M, \iota \in [0,t]}$. Then $a_{j,\iota} = \sum_{l \in I'_M} d_{l,\iota} s_{j,l}$, for $j = 1, 2$.
   D sets $\alpha_{j,\iota} = \prod_{l \in I'_M} \omega_{j,l}^{d_{l,\iota}}$. Then the simulation is carried through as described above, except that M_j for $j \notin I_M$ replace $s_{1,j} + s_{2,j}$ in the sum $s_j = \sum_{l \in I_1} s_{l,j}$ by $s_{(1,2),j}$ (recall that $\mathcal{S}$ does not even know $s_{1,j}$ or $s_{2,j}$).
2. In the simulation of $\mathcal{F}_{\text{DKG}}$, D instructs it to use the values s_j extracted from corrupt generators, and the values s_j generated by $\mathcal{S}$ in the simulation.

Concluding the Proof. If (u, v, w) is a DDH-triple, then $s_{1,l} = \log_{h_1} \omega_{1,0} = \sigma - e_2$ and $s_{2,l} = \log_{h_2} \omega_{2,0} = e_2$. Thus, $s_{1,l} + s_{2,l} = s_{(1,2),l}$, which gives $s_j = \sum_{l \in I_1} s_{l,j}$. This implies that the distribution of the output of D is identical to the distribution of $\mathcal{Z}_z(\mathcal{H}(\mathcal{A}', \pi^{(\overline{\pi}_1^{\mathcal{F}_{\text{BB}}}, \overline{\pi}_2^{\mathcal{F}_{\text{MMT}}}, \overline{\pi}_3^{\mathcal{F}_{\text{CRS}}})}))$, since all inputs to $\mathcal{A}$ during

the simulation are identically distributed to the corresponding inputs in a real execution.

If on the other hand (u, v, w) is not a DDH-triple, then $s_{1,l} = \log_{h_1} \omega_{1,0} = \sigma - e_2$ and $s_{2,l} = \log_{h_2} \omega_{2,0} = e_3$, which implies that $s_{1,l} + s_{2,l}$ is independently distributed from $s_{(1,2),l}$ for $l \notin I_M$. Since the former is used in the construction of $\alpha_{j,\iota}$ for $j = 1, 2$ and the latter is used to compute s_j, β_j and thereby y are independently distributed from $\alpha_{j,\iota}$. This is precisely the situation in the ideal model. Thus, the distribution of D in this case is identical to the distribution of $\mathcal{Z}_z(\mathcal{I}(\mathcal{S}, \overline{\pi}^{\mathcal{F}_{\mathrm{DKG}}}))$.

This implies that the DDH-assumption is broken (Definition 1), and the theorem is true. The corollary follows from Lemma 2 and Lemma 1 by use of the composition theorem of the UC-framework (Theorem 2 in Appendix A).

Acknowledgments

I thank Johan Håstad for helpful discussions in general and in particular for helping me to correct an error in a preliminary version.

References

1. M. Abe, S. Fehr, *Adaptively Secure Feldman VSS and Applications to Universally-Composable Threshold Cryptography*, to appear at Crypto 2004. (full version at Cryptology ePrint Archive, Report 2004/118, http://eprint.iacr.org/, May, 2004).
2. A. Aho, J. Hopcroft, J. Ullman, *The Design and Analysis of Computer Algorithms*, ISBN 0-201-00029-6, Addison Wesley, 1974.
3. D. Beaver, *Foundations of secure interactive computation*, Crypto '91, LNCS 576, pp. 377-391, 1991.
4. R. Canetti, *Security and composition of multi-party cryptographic protocols*, Journal of Cryptology, Vol. 13, No. 1, winter 2000.
5. R. Canetti, *Universally Composable Security: A New Paradigm for Cryptographic Protocols*, http://eprint.iacr.org/2000/067 and ECCC TR 01-24. Extended abstract appears in 42nd FOCS, IEEE Computer Society, 2001.
6. R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, *Adaptive security for threshold cryptosystems*, Crypto '99, LNCS 1666, pp. 98-115, 1999.
7. R. Cramer, V. Shoup, *A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack*, Crypto '98, pp. 13-25, LNCS 1462, 1998.
8. P. Feldman, *A practical scheme for non-interactive verifiable secret sharing*, 28th FOCS, pp. 427-438, 1987.
9. Y. Frankel, P. MacKenzie, M. Yung, *Adaptively-secure distributed threshold public key systems*, In Proc. of 7th Annual European Symposium 1999, LNCS 1643, pp. 4-27, 1999.
10. Y. Frankel, P. MacKenzie, M. Yung, *Adaptively-secure optimal-resilience proactive RSA*, Asiacrypt '99, LNCS 1716, pp. 180-194, 1999.
11. Y. Frankel, P. MacKenzie, M. Yung, *Adaptive Security for the Additive-Sharing Based Proactive RSA*, Public Key Cryptography 2001, LNCS 1992, pp. 240-263, 2001.

12. Y. Frankel, P. MacKenzie, M. Yung, *Adaptively secure distributed public-key systems*, Theoretical Computer Science, Vol. 287, No. 2, September 2002.
13. R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, *Secure Applications of Pedersen's Distributed Key Generation Protocol*, RSA Security '03, LNCS 2612, pp. 373-390, 2003.
14. R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, *Secure Distributed Key Generation for Discrete-Log Based Cryptosystems*, Eurocrypt '99, LNCS 1592, pp. 295-310, 1999.
15. R. Gennaro, M. O. Rabin, T. Rabin, *Simplified VSS and Fast-track Multiparty Computations with Applications to Threshold Cryptography*, In Proc. of the 1998 ACM Symposium on Principles of Distributed Computing, 1998.
16. O. Goldreich, *Foundations of Cryptography*, Cambridge University Press, 2001.
17. O. Goldreich, S. Micali, and A. Wigderson, *How to Play Any Mental Game*, 19th STOC, pp. 218-229, 1987.
18. S. Goldwasser, L. Levin, *Fair computation of general functions in presence of immoral majority*, Crypto '90, LNCS 537, pp. 77-93, 1990.
19. S. Goldwasser, Y. Lindell, *Secure Multi-Party Computation Without Agreement*, In Proc. of the 16th DISC, LNCS 2508, pp. 17-32, 2002.
20. S. Goldwasser, S. Micali, C. Rackoff, *The Knowledge Complexity of Interactive Proof Systems*, SIAM Journal of Computing, Vol. 18, pp. 186-208, 1989.
21. S. Jarecki, A. Lysyanskaya, *Adaptively Secure Threshold Cryptography without the Assumption of Erasure*, Eurocrypt 2000, LNCS 1807, 221-242, 2000.
22. Y. Lindell, A. Lysyanskaya, T. Rabin, *On the Composition of Authenticated Byzantine Agreement*, 34th STOC, pp. 514-523, 2002.
23. S. Micali, P. Rogaway, *Secure Computation*, Crypto '91, LNCS 576, pp. 392-404, 1991.
24. T. Pedersen, *Non-interactive and information-theoretic secure verifiable secret sharing*, Crypto '91, pp. 129-140, 1991.
25. T. Pedersen, *A threshold cryptosystem without a trusted party*, Eurocrypt '91, pp. 129-140, 1991.
26. B. Pfitzmann, M. Waidner, *Composition and Integrity Preservation of Secure Reactive Systems*, 7th Conference on Computer and Communications Security of the ACM, pp. 245-254, 2000.
27. C. Rackoff, D. Simon, *Noninteractive zero-knowledge proofs of knowledge and chosen ciphertext attacks*, 22:nd STOC, pp. 433-444, 1991.
28. D. Wikström, *A Universally Composable Mix-Net*, Proceedings of First Theory of Cryptography Conference (TCC '04), LNCS 2951, pp. 315-335, 2004.

A Review of the UC-Security Framework

In this section we give a short review of the universally composable security framework of Canetti [5]. This framework is very general, quite complex, and hard to describe both accurately and concisely. We have chosen to use a slightly simplified approach. For a general in depth discussion, intuition, and more details we refer the reader to Canetti [5]. Note that we consider only static adversaries.

Following Goldwasser, Micali and Rackoff [20] we define the parties to be interactive Turing machines, and denote the set of interactive Turing machines by ITM.

Canetti assumes the existence of an "operating system" that handles the creation of subprotocols. This is necessary to handle protocols with a large number of possible trees of calls to subprotocols, but for our purposes we may assume that all subprotocols are instantiated already at the start of the protocol.

Canetti models an asynchronous communication network, where the adversary has the power to delete, modify, and insert any messages of his choice. To do this he is forced to give details for exactly what the adversary is allowed to do. This becomes quite complex in the hybrid model. We instead factor out all aspects of the communication network into a separate concrete "communication model"-machine. The real, ideal, and hybrid models are then defined solely on how certain machines are linked. The adversary is defined as any ITM, and how the adversary can interact with other machines follows implicitly from the definitions of the real and ideal communication models.

Since each protocol or subprotocol communicate through its own copy of the "communication model", and all protocols are instantiated at the start of the protocol we need not bother with session ID:s. Such ID:s would clearly be needed if our protocols would be rewritten in the more general original security framework, but it is notationally convenient to avoid them.

We also assume that we may connect any pair of machines by a "link". Such a link is more or less equivalent to the notion of a link as defined by Goldreich [16]. Thus, the following is meaningful.

Definition 1. *An* ITM*-graph is a set* $V = \{P_1, \ldots, P_t\} \subset$ ITM *with a set of links* E *such that* (V, E) *is a connected graph, and no* P_i *is linked to any machine outside* V*. Let* ITMG *be the set of* ITM*-graphs.*

During the execution of an ITM*-graph, at most one party is* active*. An active party may deactivate itself and activate any of its neighbors, or it may halt, in which case the execution of the* ITM*-graph halts.*

The real communication model models an asynchronous communication network, in which the adversary can read, delete, modify, and insert any message of its choice.

Definition 2. *A* real communication model $\mathcal{C}$ *is a machine with a link* l_{P_i}*, to* P_i *for* $i = 1, \ldots, k$*, and a link* $l_{\mathcal{A}}$ *to a real adversary* $\mathcal{A}$*, defined as follows.*

1. *If* m *is read on* l_s*, where* $s \in \{P_1, \ldots, P_k\}$*, then* (s, m) *is written on* l_A *and* $\mathcal{A}$ *is activated.*
2. *If* (r, m) *is read on* $l_{\mathcal{A}}$*, where* $r \in \{P_1, \ldots, P_k\}$*, then* m *is written on* l_r*, and* r *is activated.*

The ideal communication model below captures that the adversary may decide if and when to deliver a message from an ideal functionality to a party, but it can not read the contents of the communication.

Definition 3. *An* ideal communication model $\mathcal{C}_{\mathcal{I}}$ *is a machine with a link* l_{P_i}*, to* P_i *for* $i = 1, \ldots, k$*, and links* $l_{\mathcal{F}}$*, and* $l_{\mathcal{S}}$ *to an ideal functionality* $\mathcal{F}$ *and an ideal adversary* $\mathcal{S}$ *respectively. Its program is defined as follows.*

1. *If a message m is read on l_s, where $s \in \{P_1, \ldots, P_k\}$, then (s, m) is written on $l_{\mathcal{F}}$ and $\mathcal{F}$ is activated.*
2. *If a message (s, m) written on $l_{\mathcal{F}}$ is returned unaltered*[1]*, m is written on l_s. If not, any string read from $l_{\mathcal{F}}$ is interpreted as a list $((r_1, m_1), \ldots, (r_t, m_t))$, where $r_i \in \{\mathcal{S}, P_1, \ldots, P_k\}$. For each m_i a random string $\tau_i \in \{0,1\}^n$ is chosen, and (r_i, m_i) is stored under τ_i. Then $((r_1, |m_1|, \tau_1), \ldots, (r_t, |m_t|, \tau_t))$, where $|m_i|$ is the bit-length of m_i, is written to $l_{\mathcal{S}}$ and $\mathcal{S}$ is activated.*
3. *Any string read from $l_{\mathcal{S}}$ is interpreted as a pair (b, τ), where $b \in \{0,1\}$ and τ is an arbitrary string. If $b = 1$ and (r_i, m_i) is stored in the database under the index τ, m_i is written on l_{r_i} and r_i is activated. Otherwise $(\mathcal{S}, \tau)$ is written to $l_{\mathcal{F}}$ and $\mathcal{F}$ is activated.*

An *adversary* can normally corrupt some subset of the parties in a protocol. A *dummy party* is a machine that given two links writes any message from one of the links on the other. There may be many copies of the dummy party. We write $\overline{P}$ for dummy parties. The ideal model below captures the setup one wishes to realize, i.e. the environment may interact with the ideal functionality $\mathcal{F}$, except that the adversary $\mathcal{S}$ has controls the communication.

Definition 4. *The* ideal model *is defined to be a map* $\mathcal{I} : \mathrm{ITM}^2 \times \overline{\mathrm{ITM}}^* \to \mathrm{ITMG}$*, where* $\mathcal{I} : (\mathcal{F}, \mathcal{S}, \overline{P}_1, \ldots, \overline{P}_k) \mapsto (V, E)$ *is given by:* $V = \{\mathcal{C}_{\mathcal{I}}, \mathcal{F}, \mathcal{S}, \overline{P_1}, \ldots, \overline{P_k}\}$ *and* $E = \{(\mathcal{S}, \mathcal{C}_{\mathcal{I}}), (\mathcal{C}_{\mathcal{I}}, \mathcal{F})\} \cup \bigcup_{i=1}^{k} \{(\overline{P_i}, \mathcal{C}_{\mathcal{I}})\}$.

If $\overline{\pi} = (\overline{P}_1, \ldots, \overline{P}_k)$, we write $\mathcal{I}(\mathcal{S}, \overline{\pi}^{\mathcal{F}})$ instead of $\mathcal{I}(\mathcal{F}, \mathcal{S}, \overline{P}_1, \ldots, \overline{P}_k)$ to ease notation. The real model is supposed to capture the properties of the real world. The parties may interact over the real communication model.

Definition 5. *The* real model *is defined to be a map* $\mathcal{R} : \mathrm{ITM}^* \to \mathrm{ITMG}$*, where* $\mathcal{R} : (\mathcal{A}, P_1, \ldots, P_k) \mapsto (V, E)$ *is given by:* $V = \{\mathcal{C}, \mathcal{A}, P_1, \ldots, P_k\}$ *and* $E = \{(\mathcal{A}, \mathcal{C})\} \cup \bigcup_{i=1}^{k} \{(P_i, \mathcal{C})\}$.

Let $(V, E) = \mathcal{I}(\mathcal{F}, \mathcal{S}, \overline{P}_1, \ldots, \overline{P}_k)$. Then we write $\mathcal{Z}(\mathcal{I}(\mathcal{F}, \mathcal{S}, \overline{P}_1, \ldots, \overline{P}_k))$ for the ITM-graph (V', E') defined by $V' = V \cup \{\mathcal{Z}\}$, and $E' = E \cup \{(\mathcal{Z}, \mathcal{S})\} \cup \bigcup_{i=1}^{k} \{(\mathcal{Z}, \overline{P}_i)\}$. We use the corresponding notation in the real model case.

A hybrid model is a mix between a number of ideal and real models, and captures the execution of a real world protocol with access to some ideal functionalities. It is also a tool to modularize security proofs. It may be viewed as if we "glue" a number of ideal and real models onto an original real model.

Definition 6. *Suppose that we are given* $(V, E) = \mathcal{R}(\mathcal{A}, \pi)$, $\pi = (P_1, \ldots, P_k)$. *Let* $(V_j, E_j) = \mathcal{I}(\mathcal{S}_j, \overline{\pi}_j^{\mathcal{F}_j})$, $\overline{\pi}_j = (\overline{P}_{j,1}, \ldots, \overline{P}_{j,k})$ *for* $j = 1, \ldots, t$, *and* $(V_j, E_j) = \mathcal{R}(\mathcal{S}_j, \pi_j)$, $\pi_j = (P_{j,1}, \ldots, P_{j,k})$ *for* $j = t+1, \ldots, s$.

We denote by $\mathcal{H}(\mathcal{A}^{(\mathcal{S}_1, \ldots, \mathcal{S}_t)}, \pi^{(\overline{\pi}_1^{\mathcal{F}_1}, \ldots, \overline{\pi}_t^{\mathcal{F}_t}, \pi_{t+1}, \ldots, \pi_s)})$ *the* hybrid model *defined as the* ITM*-graph* (V', E')*, where* $V' = V \cup \bigcup_{j=1}^{t} V_j$ *and*
$E' = E \cup \bigcup_{j=1}^{t} E_j \cup \bigcup_{i=1}^{k} \left(\{(\mathcal{S}_i, \mathcal{A})\} \cup \bigcup_{j=1}^{t} \{(P_i, \overline{P}_{j,i})\} \right)$

[1] This special rule simplifies security proofs.

Similarly as above we write $\mathcal{Z}(\mathcal{H}(\mathcal{A}^{(\mathcal{S}_1,\ldots,\mathcal{S}_t)}, \pi^{(\overline{\pi}_1^{\mathcal{F}_1},\ldots,\overline{\pi}_t^{\mathcal{F}_t},\pi_{t+1},\ldots,\pi_s)}))$ to denote the ITM-graph (V'', E'') defined by $V'' = V' \cup \{\mathcal{Z}\}$, and $E'' = E' \cup \{(\mathcal{Z}, \mathcal{A})\} \cup \bigcup_{i=1}^{k} \{(\mathcal{Z}, P_i)\}$.

Note that all real subprotocols π_j, for $j = t+1, \ldots, s$, above may be integrated into the original real protocol π. Thus a hybrid model with no ideal functionalities involved is equivalent to a real model, except that it may use several communication models.

The concept of hybrid models is generalized in the natural way, e.g. we write $\mathcal{H}(\mathcal{A}^{(A_1^{\mathcal{S}_{11}}, A_2^{\mathcal{S}_{21}})}, \pi^{(\pi_1^{\overline{\pi}_{11}^{\mathcal{F}}}, \pi_2^{\overline{\pi}_{21}^{\mathcal{F}}})})$ for a hybrid model for a real protocol that executes two subprotocols, where each subprotocol has access to a separate copy of the ideal functionality $\mathcal{F}$. Some care needs to be taken when defining the adversary for such models. If an adversary corrupts a party, it automatically corrupts all its sub-parties that are involved in subprotocols[2].

We also write $\mathcal{Z}_z$ to denote that $\mathcal{Z}$ takes auxiliary input z, and always assume that in any execution of such an ITM-graph, $\mathcal{Z}$ is activated first.

The following definition is somewhat sloppy in that we have not defined the notion of $\mathcal{M}$-adversaries rigorously, but it is a class of adversaries.

Definition 7 (Secure Realization). *Let $\mathcal{F}$ be an ideal functionality. Let $\pi = (P_1, \ldots, P_k)$, and let $\overline{\pi}_j = (\overline{P}_{j,i}, \ldots, \overline{P}_{j,i})$ be the corresponding dummy parties for $\mathcal{F}_j$, for $j = 1, \ldots, t$.*

Then $\pi^{(\overline{\pi}_1^{\mathcal{F}_1},\ldots,\overline{\pi}_t^{\mathcal{F}_t})}$ realizes $\overline{\pi}^{\mathcal{F}}$ securely with regards to $\mathcal{M}$-adversaries if for all $\mathcal{M}$-adversaries $\mathcal{A}^{(S_1,\ldots,S_t)}$ with auxiliary input $z = \{z_n\}$, $\exists \mathcal{S} \in \mathrm{ITM}$ such that $\forall c > 0$, $\exists n_0$, such that $\forall n > n_0$:
$|\Pr[\mathcal{Z}_z(\mathcal{I}(\mathcal{S}, \overline{\pi}^{\mathcal{F}})) = 1] - \Pr[\mathcal{Z}_z(\mathcal{H}(\mathcal{A}^{(S_1,\ldots,S_t)}, \pi^{(\overline{\pi}_1^{\mathcal{F}_1},\ldots,\overline{\pi}_t^{\mathcal{F}_t})})) = 1]| < \frac{1}{n^c}$.

Since the dummy parties are of no real importance we also say that π realizes $\mathcal{F}$ in the $(\mathcal{F}_1, \ldots, \mathcal{F}_t)$-hybrid model.

Canetti [5] proves a powerful composition theorem that can handle polynomially many instances of a constant number of ideal functionalities, but we only need the following weaker special case.

Theorem 2 (Composition Theorem). *Suppose that $\pi^{(\tilde{\pi}_1^{\mathcal{F}_1},\ldots,\tilde{\pi}_t^{\mathcal{F}_t})}$ securely realizes $\tilde{\pi}^{\mathcal{F}}$, and that $\pi_i^{(\tilde{\pi}_{i1}^{\mathcal{F}_{i1}},\ldots,\tilde{\pi}_{it_i}^{\mathcal{F}_{it_i}})}$ securely realizes $\tilde{\pi}_i^{\mathcal{F}_i}$, for $i = 1, \ldots, l$, with regards to $\mathcal{M}$-adversaries.*

Then $\pi^{(\pi_1^{(\tilde{\pi}_{11}^{\mathcal{F}_{11}},\ldots,\tilde{\pi}_{1t_1}^{\mathcal{F}_{1t_1}})},\ldots,\pi_l^{(\tilde{\pi}_{l1}^{\mathcal{F}_{l1}},\ldots,\tilde{\pi}_{lt_l}^{\mathcal{F}_{lt_l}})},\tilde{\pi}_{l+1}^{\mathcal{F}_{l+1}},\ldots,\tilde{\pi}_t^{\mathcal{F}_t})}$ securely realizes $\tilde{\pi}^{\mathcal{F}}$ with regards to $\mathcal{M}$-adversaries.

[2] The most general definition allows some violations of this rule.