New look at impossibility result on Dolev-Yao models with hashes

István Vajda

Abstract: Backes, Pfitzmann and Waidner showed in [7] that for protocols with hashes Dolev-Yao style models do not have cryptographically sound realization in the sense of BRSIM/UC in the standard model of cryptography. They proved that random oracle model provides a cryptographically sound realization. Canetti [9] introduced the notion of oracle hashing "towards realizing random oracles". Based on these two approaches, we propose a random hash primitive, which already makes possible cryptographically sound realization in the sense of BRSIM/UC in the standard model of cryptography.

1. Introduction

The ideal hash function paradigm was introduced by Bellare and Rogaway in 1993 ([8]), where they argued that even though results which assume an ideal hash function do not provide provable security with respect to the standard model of computation, assuming an ideal hash function and doing proofs with respect to it provides much greater assurance benefit than purely ad hoc protocol design.

In 2006, in paper [7] a related result was shown in reactive cryptographic environment stating the Dolev-Yao style abstract model cannot be realized securely in the standard model of cryptography:

In the BRSIM model in the symbolic system the trusted host (*TH*) sends an abstract hash term to the simulator (*SIM*) upon which machine *SIM* has to present a real hash value to the adversary. The main result (Theorem 1) in [7] refers to simulation failure when stating that a Dolev-Yao type ideal hash function does not have cryptographically sound realization in the sense of BRSIM/UC in the standard model of cryptography. No deterministic hash function can serve as secure realization. They showed a way for sound realization in non-standard model where hash function is implemented by random oracle. It has also been shown in [7] that collision resistance is a necessary condition to avoid simulation failure.

Ideal secrecy is a key notion in the definition of ideal hashing. Ideal secrecy means that an adversary who obtains the hash of an otherwise unknown term cannot do better than comparing this hash with self-made hashes of guessed terms. Note, even the ideal secrecy property leaks some information about the message. If we "use" random oracle for realization, it is ensured information theoretically that information will not leak, in sense that the probability of such an event will be exponentially small.

Canetti [9] introduced the notion of *oracle hashing*. He suggested the randomization of the hash value, such that different invocations on the same input result in different output, in order to restrict the possibility of gaining partial information about the hashed message by exhaustively (polynomial many times) searching the input domain.

Technical University of Budapest E-mail: vajda@hit.bme.hu

Note, oracle hashing is not a function in the message input, dislike the random oracle which is a (public) random function. In our construction, we also cannot use a function, because we want to get rid of the third party assumption and a randomized function could not be evaluated independently by different participants.

Following Canetti [9], pair *H, V* denotes the hashing and the verification algorithms, respectively. Algorithm *H,* given a security parameter *k* and input *x,* chooses a random value *r* in domain R_k and outputs a value *h.* Algorithm *V,* given *k* as well as input *h* and a guessed message*,* outputs a binary value. Let I_x denote the verification oracle. Using these notations oracle simulatability and oracle indistinguishability were defined as follows:

Oracle simulatability [9]: For any polytime adversary *B* and any polynomial $p(.)$ there exists a polytime adversary *C*, such that for any distribution ensemble $\{X_k\}$ for any polytime predicate $P(.)$ and for all large enough *k*:

$$\Pr(B(H(x,r)) = P(x)) - \Pr(C^{I_x}() = P(x)) < \frac{1}{p(k)} \quad (1)$$

where $r \in_R R_k$, and *x* is drawn from X_k.

Oracle indistinguishability [9]: For any polytime distinguisher *D* and any polynomial $p(.)$ there exists a polynomial-size family $\{L_k\}$ of sets such that for all large enough *k* and for all $x, y \notin L_k$:

$$\Pr(D(H(x,r)) = 1) - \Pr(D(H(y,r)) = 1) < \frac{1}{p(k)} \quad (2)$$

where $r \in_R R_k$.

Theorem 4 in [9] states that requirements (1) and (2) are equivalent.

Property (1) can be considered as the definition of the ideal secrecy: the task of finding information on the message with a given hash value *h* gives no more information, besides the ability of exhaustive (polynomial many times) querying verification oracle I_x.

Indistinguishability property (2) will help us to eliminate the problem of simulation failure by machine *SIM*.

Recall, wording "oracle" means that hash algorithm is run by a trusted third party: a participant sends message *x* for hashing, the oracle draws an appropriate random element *r* and outputs hash value $H(r,x)$. We want to eliminate the third party assumption, making possible for participants to calculate hash value on their own. On this way, we add the following property to the definition of ideal hashing: only those participants will be able to verify a hash value in the knowledge of the corresponding message which participants are authorized to do so by the sender of the hash value. This way an adversary trying to guess the hashed value will see a virtual oracle hashing with its ideal secrecy property. This means that we introduce a weakened ideal hash primitive, where the access to the verification algorithm is controlled by the sender of the hash value.

The structure of the submission is the following. Section 2 summarizes our contributions. Section 3 gives a short discussion on the non-standard model of [7]. In Section 4 we introduce a hash primitive, where we define the ideal and real properties of the primitive as well as the corresponding commands in the symbolic and the real system. In this section we present also our main theorem with proof sketch about the cryptographically sound realization of this primitive in the sense of BRSIM/UC in the standard model of cryptography. In Section 5 we show a construction for the primitive. Conclusions are drawn in Section 6.

2. Our contribution

We discuss the impossibility result [7] and a corresponding pseudorandom oracle model.

Our main result is the introduction of a new type of random hash primitive. The aim was to define a primitive which is collision free, provides the property of ideal secrecy, as well as a sender-controlled access to the verification algorithm. We define the ideal and real properties of the primitive as well as the corresponding additional commands in the symbolic and the real system extended by the new primitive.

Considering realization within the standard model of cryptography, in general, we see that security guaranties in the computational settings usually allow that cryptographic primitives leak partial information about their inputs. Therefore we extend the BPW's (Backes-Pfitzmann-Waidner) formalism with partial information.

Proof sketch is shown for the statement that this new type of ideal hash mapping has cryptographically sound realization in the standard model of cryptography, where the proof is carried out in the blackbox reactive simulatability (BRSIM/UC) model defined in [10]. We propose construction for a hash primitive.

3. Simulation in non-standard cryptographic model: random oracle

Fig.1. shows the overview of the symbolic system with blackbox simulation, where usual notations of BPW's approach are applied ([1-7]). Honest protocol machines are $M_{u_i}, u_i \in U$, where U stands for the set of honest identifiers.

User machine H communicates with protocol machines: initializes a new run and receives the output of the run. A secure protocol has to meet – formally defined - security requirement at the service layer between the user machine H and the honest protocol machines.

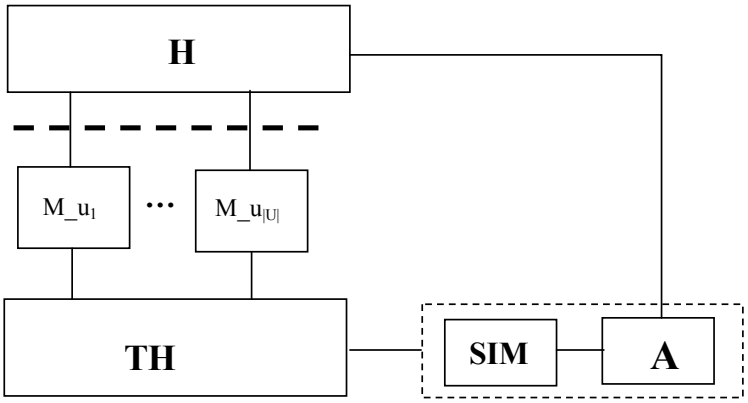

Fig. 1.: Overview of the symbolic system

In this model, the adversary is allowed to initialize new protocol runs via user machine *H*. Trusted host *TH* is an important element of the symbolic system. Protocol machines and the adversary communicate with each other via machine *TH*. All cryptographic primitives are moved from protocol machines into *TH* and are available for participants by sending commands to *TH*. Machine *TH* contains also a database, which stores the history of all operations carried out by protocol machines in cooperation with *TH*. User machine H initializes new protocol runs by sending appropriate input to a protocol machine M_u_i and at the end of the run the same machine will report the result to machine *H*, via the service interface.

In paper [7], in the symbolic model the random oracle was included in machine *SIM*, in the real system it was a third party with public access. In our symbolic system the verification oracle I_x is part of the trusted host, the random hash primitive is included in SIM, while in the real system each participant (protocol machines and the adversarial machine) has its own random hash primitive.

When in the symbolic system the trusted host sends an abstract hash term to the simulator (*SIM*), the simulator has to present a real hash value to the adversary. In case of successful simulation the simulated value must be indistinguishable from the true one in the view of the adversary. In case of usual hash functions simulation failure occurs with overwhelming probability if the adversary gets access also to the true hashed message (e.g. directly from the user (*H*)), because in this case the adversary is able to verify the correctness of the simulated hash value.

The corresponding application scenario is when a participant commits to a message by sending its hash value, where the message is revealed only sometime later. If the simulator has to simulate a bitstring for the true hash value before knowing the message, then whatever it picks will most likely not match the message. The ***commitment problem*** based on hashing techniques is kept in mind during this paper (see constructions in Section 5).

The impossibility result in [7] has an important practical benefit: it tells us that if we want to have a proof of a protocol with hashes in the random oracle model, then instead of usual cryptographic proofs (which are by hand proofs, practically, can be carried out only for small protocols) we can conduct the proof on the symbolic

version of the protocol where the hash functions are substituted by the Dolev-Yao style abstract model.

Note also, *the problem with the random oracle is not that it assumes exponential complexity, but the fact that we need a trusted third party to run the hash function.* Indeed, this third party can be implemented with polynomial complexity, if we substitute the random function by a pseudorandom function (PRF) with appropriate dimensions (Fig.2.).

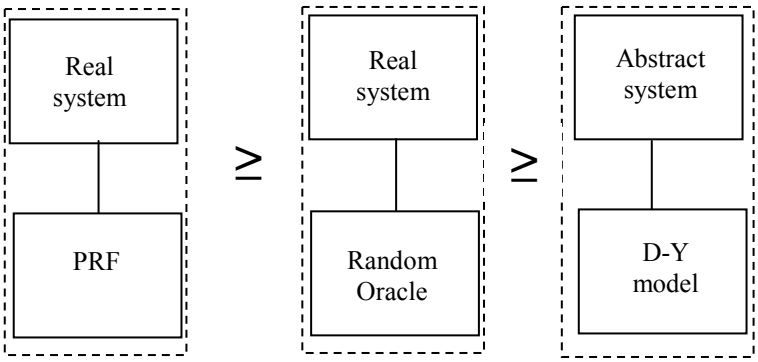

Fig.2. Sound realization of Dolev-Yao style ideal hash function by a PRF
("≥" signs "as secure as" in the actual meaning: comp and ROM, respectively)

Let $f(x, y)$ be a public deterministic function, where the random kernel and the message to be hashed are substituted into argument *x* and *y*, respectively. When a participant asks the third party for the hash of a message *m* for the first time, the third party chooses a fresh random string *r* as the kernel and outputs the following hash value:

$$h = f(r, m) \tag{3}$$

as well as it stores triplet $\{m, r, h\}$.

Theorem 1: Abstract (Dolev-Yao style) hash functions can be realized securely by a third party with only polynomial complexity in BRSIM/UC model.

Proof: By definition, there does not exist any efficient algorithm which can distinguish a PRF from a random function. Therefore, there cannot be a distinguishable view at the service interface, because the proof system itself would provide a distinguishing algorithm. □

We step further on this way. We will use efficient hash primitive, however, we want to eliminate the assumption of a third party. In the next section we introduce such a hash primitive.

4. A random hash primitive in the standard model of cryptography

4.1. Symbolic system

We introduce a hash primitive (*r_hash*) with ideal collision freeness and ideal secrecy property as well as ideal sender-controlled verification property in the symbolic system. Subsequently we define the real properties of the primitive as well as the corresponding commands in the symbolic and real system.

Definition 1: The ideal properties of *r_hash* primitive are the following:

i.) Ideal collision freeness: $r\_hash(m) = r\_hash(m') \rightarrow$ m= m′ *for all* m, m′.

ii.) Ideal secrecy: If we have a new hash value *h* of message *m* and want to find information about the hashed message we cannot do better than forgetting about *h* and just relying on the verification oracle I_m by invoking it with guessed messages polynomial many times.

iii.) Verification capability: Only those users are able to verify (re-calculate) a hash value in the knowledge of the corresponding message, who are included in a set (*Vset*). *Vset* is determined by the sender of the hash value.
□

Attacking ideal secrecy means that the attacker wants to collect information about the message, with the aim to identify the message: in formal description, to get a handle to the corresponding entry in the database *D* of *TH*.

Each entry in *D* has attributes:

$$(ind, type, \arg, hnd_{i_1}, ..., hnd_{i_n}, hhnd_{j_1}, ..., hhnd_{j_m}, phnd_{k_1}, ..., phnd_{k_l}, len, p\_\inf)$$

where new type of handlers, *hhnd* (hit handler) and *phnd* (partial information handler) as well as an additional argument *p_inf* (partial information) is introduced (defined subsequently).

Below we give the additional commands for communication with the trusted host (*TH*) in the symbolic system, as well as with the protocol machines in the real system. The command set is adapted from [7].

Commands: The trusted host extended by $r\_hash()$ accepts the following additional commands at every port $in_u?$:

r_hashing: $h^{hnd} \leftarrow r\_hash(l^{hnd}, vset^{hnd})$.
Let $l := D[hnd_u = l^{hnd} \wedge type = list].ind$, $vset := D[hnd_u = vset^{hnd} \wedge type = list].ind$, $length := r\_hash\_len^*(k)$ and return $\downarrow$ if $l = \downarrow$ or $vset = \downarrow$ or $length > \max\_len(k)$.

Let $h := D[type = 'r\_hash' \wedge \arg[1,2] = (l, vset)].ind$ .If $h \neq \downarrow$ then $h^{hnd} := ind2hnd_u(h)$ else set $h^{hnd} := curhnd_u ++$ and

$D :\Leftarrow (ind := size++, type := 'r\_hash', \arg := (l, vset), hnd_u := h^{hnd}, len := length)$

preimage test: $b \leftarrow is\_hash\_of(l^{hnd}, vset^{hnd}, h^{hnd})$.

Let $l := D[hnd_u = l^{hnd} \wedge type = list].ind$, $vset := D[hnd_u = vset^{hnd} \wedge type = list].ind$, $h := D[hnd_u = h^{hnd} \wedge type = 'r\_hash'].ind$ and return $\downarrow$ if $l = \downarrow$ or $vset = \downarrow$ or $h = \downarrow$.

If $D[h].\arg[1,2] = (l, vset)$ return $b := true$ else $b := false$. If $u \notin Vset$ and $b := true$ set $h^{hhnd} := ind2hhnd_u(h)$ else if $u \notin Vset$ and $b := false$ set $h^{phnd} := ind2phnd_u(h)$, $p\_\inf := ()$.

Execution of command $is\_hash\_of()$ in the *preimage test* branches: the participant which sends the command to the trusted host:

a.) is not in *Vset*:

a1.) if by a guess the participant hits the preimage, it gets a *hhandle* (hit handle) to the hash entry, set by algorithm $ind2hhnd_u(h)$, otherwise

a2.) it will get a *phandle* (partial information handle) to the hash entry set by algorithm $ind2phnd_u(h)$,

where *h* is the index of the hash entry in the database of TH.

b.) it is in *Vset*: no extra handles are set.

Partial information is stored as data by the participant represented by blank argument (*p_inf* is set to ( ).). This way partial information is taken into account in the symbolic model, by adding label to the hash entry, which specifies that the corresponding message is known, unknown or partially known by a participant without providing details about information.

Using algorithm $ind2phnd_u$ trusted host *TH* determines a *phandle* for *u* to an entry $D[i]$ in its database:

if $D[i].phnd_u = \downarrow$ it sets $D[i].phnd_u := curphnd_u ++$ else let $ind2phnd_u(i) := i$.

Definition of algorithm $ind2hhnd_u(h)$ is analogous.

Similarly as in [7], the trusted host is parametrized with length functions, because length leaks to the adversary and because higher protocols may need to know the length of certain terms for honest participants.

$r\_hash\_len(k)$: output length of the real hash primitive, which is polynomial in security parameter *k*,

$r\_hash\_len^*(k) := list\_len(len('r\_hash'), r\_hash\_len(k))$,
$\max\_len(k)$: polynomial bound on the length of messages in the system.

Local adversary commands for r_hash:
Commands *Generate unknown hash* and *Parameter retrieval*, similar to the corresponding commands in [7].

4.2. Real system

In the real system each entry in database D_u of protocol machine *u* has attributes:

$((hnd_u, hhnd_u, phnd_u), word, type, add\_\text{arg})$

where the handler is a usual handler (hnd_u), hit handler ($hhnd_u$) or partial information handler ($phnd_u$), furthermore partial information is stored in *add_arg* (detailed subsequently).

First we define the real *r_hash* algorithm. Next the commands follow for its use in the real system. Finally, we present the theorem which states that the real *r_hash* primitive is a cryptographically sound realization of the ideal primitive of Definition 1.

Definition 2: Algorithm $r\_hash(m, Vset)$ is defined as follows:

$r\_hash: \{0,1\}^* xV \rightarrow$ r.v. over $\{0,1\}^{r\_hash\_len(k)}$

is a random mapping over the message space, where V is the power set of U. It can be evaluated efficiently and has the following properties:

i.) collision free
ii.) random variables $r\_hash(m_1, Vset)$ and $r\_hash(m_2, Vset)$ are indistinguishable for any $m_1 \neq m_2$ and any $Vset \in V$.

We specialize this hash primitive by introducing auxiliary function

$Rhash(r, m, Vset): \{0,1\}^{rand\_len(k)} x\{0,1\}^* xV \rightarrow \{0,1\}^{r\_hash\_len(k)}$

such that if we substitute random value into parameter *r* we get $r\_hash(m, Vset)$.

The verification algorithm $Ver(m, h, id)$ is defined as follows:

$Ver: \{0,1\}^* x\{0,1\}^{r\_hash\_len(k)} xID \rightarrow \{0,1,\downarrow\}$

where the inputs are the following, in order: hash value (h), message (m), user identifier (id). The evaluation of the output is the following. First, algorithm

$$r\_decrypt(h,id) \rightarrow \{r,\downarrow\}$$

is called, which outputs $\downarrow$ if $id \notin Vset$, else it outputs r.

$Ver()$ outputs $\downarrow$, if $r\_decrypt()$ outputs $\downarrow$. Otherwise, the output is 1 if $h = Rhash(r,m,Vset)$, else it is 0.
□

In order to emphasize that a hash value is the hash of the message, we use also notation $r\_hash_{Vset}(m)$ for $r\_hash(m,Vset)$.

Corollary of Theorem 4 of [9]: Hash primitive by Definition 2 meets requirement of ideal (oracle) secrecy.
Proof: Straightforward, according to the equivalence of requirements (1) and (2). □

Property iii.) in Definition 2. formalizes the limitation of the verification capability: when a user sends an $r\_hash_{Vset}(m)$ hash value to a set (*Vset*) of users, any user within this set will be able to verify the hash value assumed it has access also to the corresponding message *m*. The actual value of the random parameter is available only to a set of users selected by the sender of the hash value.

Constructors and destructors for r_hash

r_hash constructor: $h^* \leftarrow make\_r\_hash(l,vset)$.
Let $h \leftarrow r\_hash(l,vset)$ and return $h^* := ('r\_hash', h)$.

Rhash constructor: $H^* \leftarrow make\_Rhash(r,l,vset)$.
Let $H \leftarrow Rhash(r,l,vset)$ and return $H^* := ('Rhash', H)$.

$r\_decryption: rnd \leftarrow r\_decryption(h^*,id)$
If $id \in Vset$, return $r := r\_decrypt(h,id)$ else $\downarrow$.

The execution of command $is\_hash\_of()$ is a little different in the real case compared to the ideal case: it may happen that a participant (adversary) is able to show up a preimage of a hash value, by collecting an amount of partial information, which makes possible for him successful guessing, not just in case of a random hit.

In the real system when guessing a message self made hashes are compared to the targeted hash value.

Partial information is stored as data by the participant represented in binary form (e.g. the corresponding predicate is represented). It is assumed that data corresponding to partial information is never transmitted (it is used to improve further guesses). Partial information is accumulated as a sequence of trials is carried out. The accumulation of information is represented by a list of predicates. This modeling leads to a formal accumulation algorithm $add\_p\_\inf$ which appends the new predicate to the list $p\_\inf$ placed in $add\_\arg$. The input of this algorithm is $(l, vset, h)$, the output is the lengthened list $p\_\inf$.

Commands in the real system model

Protocol machine M_u extended by $r\_hash()$ accepts the following additional commands from user machine H at port $in_u?$:

r_hashing: $h^{hnd} \leftarrow r\_hash(l^{hnd}, vset^{hnd})$
If $D_u[l^{hnd}].type \neq list$ or $D_u[vset^{hnd}].type \neq list$ return $\downarrow$. Otherwise set $l = D_u[l^{hnd}].word$, $vset = D_u[vset^{hnd}].word$ and $h^* \leftarrow make\_r\_hash(l, vset)$. If $|h^*| > \max\_len(k)$ return $\downarrow$, else $(h^{hnd}, D_u) := (h^*, r\_hash, ())$.

preimage test: $b \leftarrow is\_hash\_of(l^{hnd}, vset^{hnd}, h^{hnd})$
If $D_u[l^{hnd}].type \neq list$ or $D_u[vset^{hnd}].type \neq list$ or $D_u[h^{hnd}].type \neq 'r\_hash'$ return $\downarrow$.
Otherwise, set $l := D_u[l^{hnd}].word$, $vset := D_u[vset^{hnd}].word$, $h := D_u[h^{hnd}].word$,
if $u \in Vset$, $r := r\_decryption(h^*, id)$ and $H^* := make\_Rhash(r, l, vset)$. If $h = H^*$ set $b := true$ else $b := false$,
if $u \notin Vset$, set $h^* := make\_r\_hash(l, vset)$.
If $h = h^*$ set $b := true$ and set $h^{hhnd} := ind2hhnd_u(h)$.
If $h \neq h^*$ set $b := false$ and set $h^{phnd} := ind2phnd_u(h)$,
$p\_\inf := add\_p\_\inf(l, vset, h)$.

Theorem 2: The symbolic system with ideal *r-hash* model (Definition 1.) is securely implemented in the real system with the real *r-hash* primitive (Definition 2.) in the sense of BRSIM/UC in the standard model of cryptography, assumed that honest users authorize only honest users to carry out verification.

Proof (Sketch): The proof technique in [7] can be applied to our case. Differences are the following:

i.) simulation of real hash values from the abstract terms by the simulator,

ii.) negligibility of the probability of *Nonce-Coll* event, which contains runs where collisions of random elements (in hash primitive) happened (at "word uniqueness" invariant in [7]),

iii.) negligibility of the probability of *Nonce_Guess* event, which contains those runs in which the adversary guessed a random string that he had no information about (at "word secrecy" invariant in [7]).

i.) The key question is the simulation of the real version of an abstract *r_hash* term h^i sent by trusted host (*TH*) to the simulator (*SIM*). In order to be successful, the simulator has to produce a simulated *r_hash* value h' the distribution of which is (computationally) indistinguishable from the distribution of the true *r_hash* value h^r (in the view of the adversary).

The idea is that instead of the true hash value $H_{Vset}(m)$, the simulator will produce $H_{Vset}(1^{|m|})$, the hash of constant message $1^{|m|}$. A polynomial adversary is not able to distinguish $H_{Vset}(m)$ and $H_{Vset}(1^{|m|})$, assumed the adversary is not included in the corresponding *Vset*.

ii.) A run is put into the set *Nonce_Coll*, when a (one time) random element equals a previous value in an r_hash entry. Therefore probability of *Nonce_Coll* event is negligible by polynomial complexity of all elements of the system.

iii.) If a successful guess of preimage happens, the result (guessed message) may be passed to the user machine (*H*) and such an event may make difference in the views of the two systems (symbolic and real). (Recall, the view of the user comprises the information flow through the service interface as well as the state of machine H.)

The difference in the run of the real and symbolic system with respect to successful guess of preimage, is that in the real system successful guess of preimage may happen (in principle) also with use of accumulated partial information (event $Parc\_real$). The probability of the event when the views in the two system differ ($Viewdiff\_guess$) can be upper bounded by the following way:

$$P(Wievdiff\_guess) \le P(Hit\_ideal) + P(Hit\_real) + P(Parc\_real) \qquad (4)$$

By the Corollary if we consider the success of guessing any given partial information (characterized by a predicate) in the knowledge of the hash value we can consider equivalently the guessing of the same partial information by having access only to the

verification oracle I_x (i.e. by formula (1) the difference of success probabilities is negligible). From the size of the message domain (by uniform a priori distribution) and the assumed polynomial complexity bounds, it follows that hitting a preimage based on querying oracle I_x is exponentially small. Hence, considering upper bound (4) is negligibly small, because

$$P(Wievdiff\_guess) \le \varepsilon_1 + \varepsilon_2 + \varepsilon_3$$

where $\varepsilon_1, \varepsilon_2$ are exponentially small and ε_3 is negligible.

□

Necessity of the assumption in Theorem 2:
Here follows an informal argument to the necessity of the assumption in Theorem 2: The hashing algorithm must be random. If the participants do hash calculation on their own (i.e. without a trusted third party), then the random values used by the hashing algorithm must be transmitted together with the hash value, assumed the intended recipients have to be able to verify the hash value. If the adversary has also access to these random values, he can also carry out the verification, which may result in simulation failure. Therefore, the only way to reconcile all these requirements is to limit the verification capability to a set of honest users.

5. Constructions

The construction of oracle hashing (Canetti [9]) gave the initializing idea for our construction.

Properties i-iii.) in Definition 2. are built into the primitive one by one:

(i.) We start from a collision free hash function $h(x)$.

(ii.) We need a random function $F(r,x)$, where x is the input and r is a random parameter, which function provides the indistingushability property in input x.

(iii.) We assume, function $F(r,x)$ has the following parsing property:

$$F(r,x) = r, \hat{F}(r,x), \tag{5}$$

where $\hat{F}$ is one way in both inputs r and x. Random parameter r, appearing as the first term on the RHS of (5) is protected in order to be revealed only for intended users (*Vset*).

Constructions for $F(r,x)$:

Construction 1. For oracle hashing Canetti [9] gave the following construction over group G, where the DDH (Diffie-Hellman Decision) problem is hard:

$$F(r,x) = r, r^x$$

where $r \leftarrow_R G$. It was shown in [9] that this construction meets (equivalent) requirements (1) and (2).

Construction 2. Consider the following construction over group *G*:

$$F(r,x) = r, p^r x$$

where $r \leftarrow_R S$, $S = \{1,2,...,|G|\}$, furthermore p is a randomly chosen element from *G*, publicly known in the system.

Proof of Construction 2:
The concept of transforming a chosen pair of "inputs" into indistinguishable "outputs" is a well known concept in provable security of encryption transformations, which is called IND-CPA security. Construction 2. is based on ElGamal public key encryption transformation over *G*, which provides IND-CPA security. Recall, in ElGamal encryption transformation:
$pk = g^z (= X),\ sk = z,\ E_{pk}(m) = (g^y, X^y m)$,
where $z \leftarrow_R S$, $y \leftarrow_R S$ and g is a generator of *G*. In Construction 2. we keep only the second part of the ciphertext with $g = X$, $r = y$, headed by *r*. Note, we keep the indistinguishability property. The one way properties of the corresponding mapping $\hat{F}(r,x)$ also follow. □

Example (*commitment problem*): **P**articipant *u* commits to message *m* by sending $\hat{F}(r,h(m))$. When later on message *m* is revealed for a set of participants given by *Vset*, participant *u* sends

$$m, \{\tilde{E}_{pk_1}(r), ..., \tilde{E}_{pk_{Vset}}(r)\}$$

where
$pk_1, ..., pk_{Vset}$: public keys of users in *Vset*
$\tilde{E}_{pk_i}(r)$: is the encryption of "message" *r*, where $\tilde{E}_{pk}()$ is a trapdoor one way permutation.
□

Construction for $H_{Vset}(m)$:

Using the above notations:

$$H(r,m,Vset) = \hat{F}(r,h(m)), \{\tilde{E}_{pk_1}(r), ..., \tilde{E}_{pk_{Vset}}(r)\} \qquad (6)$$

Let $H_{Vset}(m)$ denote hash value when parameter r is drawn randomly in function $H(r,m,Vset)$.□

Theorem 3: Construction $H_{Vset}(m)$ by Definition 3. is a realization of hash primitive $r\_hash_{Vset}(m)$ by Definition 2.

Proof:
i.) Collision freeness of $h(x)$ is retained by the encryption.
ii.) Mapping $\hat{F}(r,h(m))$ provides indistinguishability in message *m*:
We can not distinguish message pair m,m' if we cannot distinguish the corresponding pair of hash values $h(m), h(m')$. Indeed, if we could distinguish message pair m,m', then we could distinguish the pair of hash values $h(m), h(m')$, because $h(x)$ is assumed to be collision free.
Furthermore $\tilde{E}_{pk_1}(r),...,\tilde{E}_{pk_{Vset}}(r)$ is a random vector independent of the message.
iii.) Random nonce *r* can be revealed just by participant in *Vset*, because of the assumed property of encryption $\tilde{E}_{pk}()$, furthermore because mapping $\hat{F}(r,x)$ is assumed to be one way in input *r*.
□

Construction (6) seems too complex compared to usual hash functions. It could be considered first of all as a theoretical construct, a provably secure construction for a commitment scenario (Example above), where commitment is made by sending only $\hat{F}(r,h(m))$ (Construction 1. or 2.), and the encryptions are sent only when the message is to be revealed.

As for a possible implementation, the number of extra operations, the trapdoor one way permutations depends on the size of *Vset* (we could imagine a typical size of one or two, where in the latter case it is sent also to a third party). Applying ElGamal encryption provides the advantage of homogenious arithmetic (i.e. working over the same group *G*). If RSA encryption is applied, we have be careful with the selection of public keys if $|Vset| \geq 3$ to avoid the well know problem of small exponenst.

6. Conlusion

Hash functions are important cryptographic primitives when one way and/or collision free transformation is needed. The usual approach of deterministic hash functions fails to provide a provably secure realization of ideal properties formalized in the corresponding Dolev-Yao model. By result [7] random oracles provide cryptographically sound realization in the sense of BRSIM/UC in non-standard model of cryptography. Here follows that we have to modify the expectations against ideal hashing if we want cryptographically sound realization in the standard model of cryptography. In this paper, we proposed a weaker ideal hash primitive to obtain cryptographically sound realization in the standard model of cryptography, where the weakening means restricted access to efficient verification by participants.

References

[1] M. Backes, B. Pfitzmann, and M. Waidner. A universally composable cryptographic library. *IACR Cryptology ePrint Archive*, Report 2003/015, http://eprint.iacr.org/, January 2003.

[2] M. Backes and C. Jacobi. Cryptographically sound and machine-assisted verification of security protocols. In *Proc. 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 2607 of *Lecture Notes in Computer Science*, pages 675–686. Springer, 2003.

[3] M. Backes and B. Pfitzmann. A cryptographically sound security proof of the Needham-Schroeder_Lowe public-key protocol. *Journal on Selected Areas in Communications*, 22(10):2075–2086, 2004.

[4] M. Backes and B. Pfitzmann. A General Composition Theorem for Secure Reactive Systems. *Theory of Cryptograpy Conference (TCC 2004)*, *LNCS 2951*, pp. 336-354, 2004.

[5] M. Backes and B. Pfitzmann. Cryptographic key secrecy of the strengthened Yahalom protocol via a symbolic security proof. Research Report 3601, IBM Research, 2005.

[6] M. Backes, I. Cervesato, A.D. Jaggard, A. Scedrov and J-K. Tsay. . A cryptographically sound security proof for Basic and Public key Kerberos. Computer Security – *ESORICS 2006*, LNCS, Volume 4189/2006, 362-383.

[7] M. Backes, B. Pfitzmann, and M. Waidner. Limits of the Reactive Simulatability/UC of Dolev-Yao Models for Hashes. *Cryptology ePrint Archive: Report 2006/068.* also in *Workshop on Formal and Computational Cryptography (FCC 2006) (2006)*

[8] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. *First Annual Conference on Computer and Communications Security*, pages 62-73, Fairfax, 1993. ACM.

[9] R.Canetti. Towards Realizing Random Oracles: Hash Functions That Hide All partial Information. *Advance in Cryptology – CRYPTO '97*, LNCS 1294., pp.455-469, 1997.

[10] B. Pfitzmann and M. Waidner. Composition and integrity preservation of secure reactive systems. In *Proc. 7th ACM CCS*, pages 245–254, 2000.

[11] I.Vajda. Cryptographically Sound Security Proof for On-Demand Source Routing Protocol EndairA. *Cryptology ePrint Archive Report 2011/103*. http://eprint.iacr.org/2011/103.pdf

[12] I.Vajda. Framework for Security Proofs for Reactive Routing Protocols in Multi-Hop Wireless Networks. *Cryptology ePrint Archive Report 2011/237*. http://eprint.iacr.org/2011/237.pdf