T-79.159 Cryptography and Data Security

Lecture 6: Authentication

Helger Lipmaa
Helsinki University of Technology
helger@tcs.hut.fi

Recap

- Until now, we have been mostly talking about *confidentiality*: how to keep data secret
- Two problems with the secret key cryptography: key distribution and authentication (with non-repudiation)
- Key distribution: Diffie-Hellman and derivatives
- Topic of today's talk: identification and authentication

How to prove that you are who you are?

- Prove that you *own* something

  Classically: passport, driver license, key

- Prove that you *know* something

  Classically: password

- Prove that you *are* something

  Semiclassically: biometrics, picture

Cryptographic approach

- Remember, we talk about digital communication

- Proving that you *are* something almost impossible (biometrics is often deceiving, and it is not our area)

- Proving that you *own* something: ok, but own what?

  ⋆ Own a book with passwords? This is then proving that you know something (passwords!)

- Proving knowledge: this is cryptographic approach

Major concept: Proofs of knowledge

- Intuition: you are P if you know her secret key. You prove the knowledge of this secret to the verifier

- All possible verifiers V know the public key, and can verify the proof, based on that

- Security criterion 1:

$$\Pr[V \text{ accepts } P\text{'s proof}] = \begin{cases} 1 - \varepsilon \ , & P \text{ knows secret} \\ \varepsilon \ , & P \text{ does not know secret} \end{cases} .$$

- Criterion 2: After (possibly many) interactions with a prover, V should not be able to imposter her

Nontransferability vs Nonrepudiation

Identification: You identify yourself as Peggy P, by proving you know her secret. Verifier V must not be able to replay your role with some other verifier (*nontransferability*)

Authentication: You bind some data to yourself, so that the verifier can later prove to others that this document was authenticated by you (you cannot repudiate signing: *nonrepudiation*).

Nonrepudiation $\neq$ Nontransferability!

- MACs made it possible to have authentication without nonrepudiation

Signatures: shortly

- You must authenticate some data m as coming from you
  - ⋆ Everybody can see that it is from you
- Usage example: legal documents
  - ⋆ Signature must be binding
  - ⋆ You may get sued based on your signature. Several countries have digital signature laws
- We will touch practical aspects in a later lecture (in particular how to bind you with your secret)

Signatures: shortly

- Signing: a mathematical function of the data m and Alice's secret key secret sk_A,

$$s = \mathsf{sign}(sk_A, m) \quad .$$

- Verification: function that accepts if s was signed by Alice:

$$\boxed{s = \mathsf{sign}(sk_A, m) \text{ if and only if } \mathsf{ver}(pk_A, m, s) = 1}$$

- Initial idea (1975–1980): For any public key cryptosystem, use its secret key for signing and the public key for verification

RSA signature scheme

- Public key: (e, n), $n = pq$, where p, q are large primes and e is a public exponent

- Secret key: (p, q, d), where d is the secret exponent

- Signing m: $s = m^d \mod n$

- Verification: Check whether $m \stackrel{?}{=} s^e \mod n$

- Bad! We will see later, why

Identification protocols: idea (1/2)

- A proves her identity to B
- A must know the secret, it is not sufficient if she replays an old session
  - ⋆ Cannot be achieved if B's actions are deterministic
- B must *not* be able to replay the protocol to C to pretend being A
  - ⋆ Cannot be achieved if A's actions are deterministic
- Thus, an identification protocol must include some randomness, supported by both A and B

Identification protocols: idea (2/2)

- To have mutual randomness, A must send a message that depends on B's random coins, and the same for B

- General idea, challenge-response:
  - $\star$ A sends a random-looking element to B,
  - $\star$ B challenges A with a random message,
  - $\star$ A responds with a message that shows that she knows the secret

- Thus, *interactivity* is needed

Randomness and interactivity

- **Very important:** randomness and interactivity are needed to achieve many cryptographic goals

	Signing	Encryption	Identification
Randomness	No*	Yes	Yes
Interactivity	No	No	Yes

* Many signature schemes still use randomness (only in a very few settings it is known how to make deterministic and yet secure signature schemes)

Identification protocols: usage scenarios

- Smart doors: use smartcard to get in
- ATM: identify yourself as a legal customer
- Different websites, e-banking
- Etc

Common problem: must avoid re-execution of the protocol by somebody else

3-round proofs of knowledge: history

- The first known three-move (challenge-response) proof of knowledge is by Fiat and Shamir (based on the difficulty of factoring)

- . . . extended later by Fiat, Feige and Shamir (1988) and finally by Feige and Shamir (1990) that defined the notion of “witness hiding”.

- Other desirable objectives of identification protocols are: special honest-verifier zero-knowledge, collision intractability, proofs of knowledge, special soundness. A *witness hiding proof of knowledge* can be used as a secure identification scheme.

Some more 3R identification protocols

1988 — Guillou-Quisquater

1990 — Ong-Schnorr (witness hiding, !PoK, CI, !SS)

1991 — Schnorr (SS, SHVZK, PoK)

1992 — Brickell-McCurley (witness hiding)

1992 — Okamoto-Schnorr (witness hiding)

Notation

- If $\mathfrak{A}$ is an algorithm, then the notation

  $$a \leftarrow \mathfrak{A}(b)$$

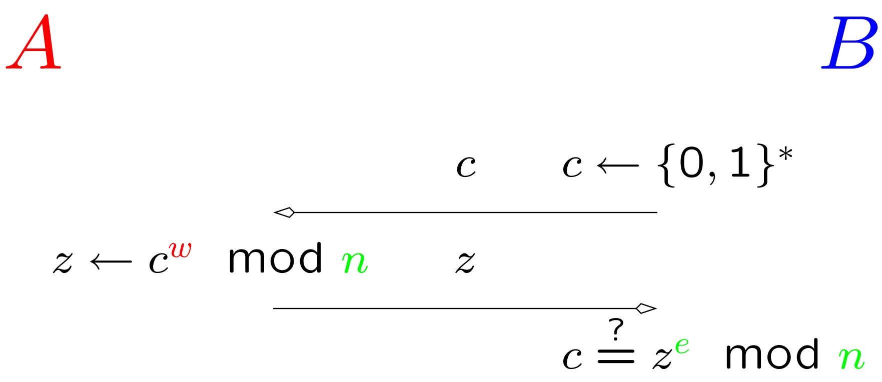
  refers to the computation of the output “a”, on input bit string “b”.

- If V is a set, $v \leftarrow V$ denotes uniform and random selection of an element v from V.

- Red variables are known only to A. Blue variables are known only to B, green variables are known to both from the start of the protocol

Faulty first idea for protocol

- Use something like RSA-based authentication, where w (*witness*) is the secret key of A and e is the corresponding public key, and c is a random challenge:



A

B

$c \leftarrow \{0,1\}^*$

c

$z \leftarrow c^w \mod n$

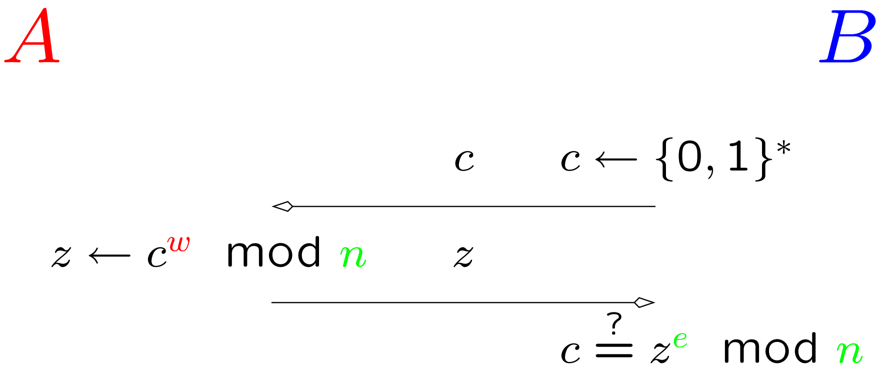
z

$c \overset{?}{=} z^e \mod n$

This prevents A from replaying the protocol.

Still bad. Why?

Faulty first idea for protocol



A $\qquad$ B

c $\qquad$ $c \leftarrow \{0,1\}^*$

$z \leftarrow c^w \mod n$ $\qquad$ z

$c \overset{?}{=} z^e \mod n$

Weakness: the signed texts are chosen solely by B, and this may allow the verifier to mount chosen-text attacks.

Σ-Protocols. General Setting (1/2)

- Σ-*protocol* is a three-move protocol between two parties, “prover” A and “verifier” B, where the prover acts first.

- The prover and verifier are modeled as probabilistic polynomial time interactive Turing machines.

- Furthermore, a honest verifier is expected to send only uniformly and randomly chosen bits.

Σ-Protocols. General Setting (2/2)

- Such protocol is denoted by (A, B).

- Then we say that (A, B) is a Σ*-protocol for relation* R.

Σ-Protocols. Example

- Secret key is w, public key is $v = g^w$

- Then $R(v, w) = 1$ iff $v = g^w$

- We need a Σ-protocol for proving that A knows w, s.t. $R(v, w) = 1$, or equivalently, such that $g^w = v$

Σ-Protocols. Inputs (1/2)

- Both principals know v (the *public key* of A)
- Only A knows w (the *secret key*/*witness* of A)
- R_A [resp R_B] is the random *secret* input of A [resp B].
  - ⋆ Recall that randomness was necessary

Σ-Protocols. Inputs (2/2)

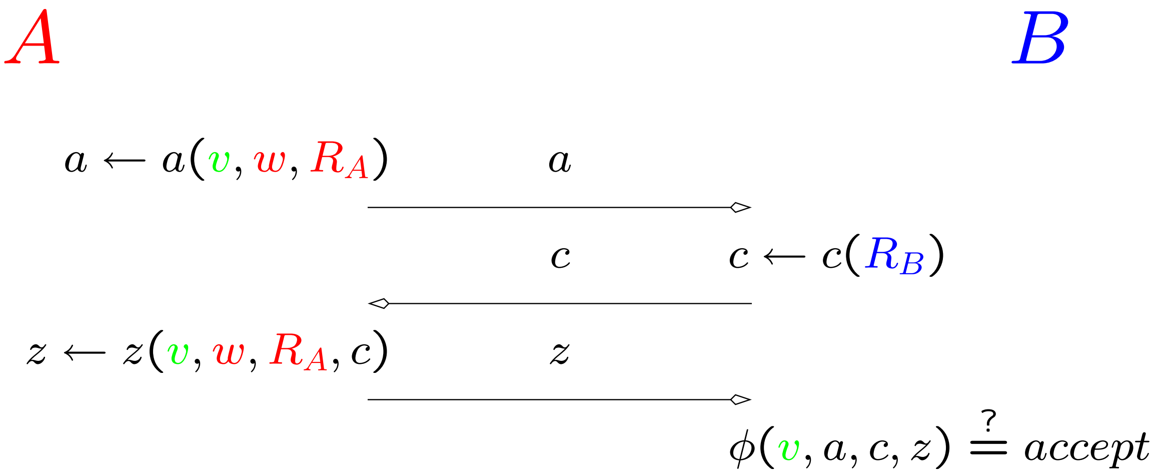
- The pair $(v, w) \in R$, where $R \subset \{0,1\}^* \times \{0,1\}^*$ is a publicly known, typically (but not necessary) efficiently verifiable relation. Let

$$R_W(v) := \{w : (v, w) \in R\} \quad \text{and}$$
$$R_X := \{v : R_W(v) \neq \emptyset\} \ .$$

- Intuitively: $R_W(v)$ is the set of secret keys corresponding to public key v, and R_X is the set of secret keys that have a corresponding public key.

- Simplified presentation: all secret keys have a public key, i.e., R_X is the set of public keys. (For some well-known schemes like Guillou-Quisquater, this is not the case!)

Σ-Protocols. Description



A B

$a \leftarrow a(v, w, R_A)$ $\xrightarrow{a}$

$\xleftarrow{c}$ $c \leftarrow c(R_B)$

$z \leftarrow z(v, w, R_A, c)$ $\xrightarrow{z}$

$\phi(v, a, c, z) \stackrel{?}{=} accept$

a: *initial message*. $t_A = |a|$ is the *authentication length* — PPT algorithm
c: *challenge*, $c \leftarrow \{0, 1\}^{t_{R_B}}$.
z: *reply* (may reuse a) — PPT algorithm.

Finally, B invokes a polynomial time computable predicate ϕ to check whether the *conversation* (x, a, c, z) is *accepting*.

Recall: Discrete Logarithm Problem, Syntax

- Let G_q be a group of prime order q. Let $g \in G_q$, $g \neq 1$. For each $h \in G_q$ there is a unique $w \in \mathbb{Z}_q$ such that $g^w = h$. w is called the *discrete logarithm* of h wrt g.

- Let $\mathcal{G}$ be a family of groups of prime order such that the group operations can be performed efficiently, group elements can be efficiently sampled with uniform distribution and group membership as well as equality of group members can be efficiently tested.

Recall: Discrete Logarithm Problem, Semantics

- Let $\mathfrak{Gen}$ be a PPT *generator algorithm* that on input 1^k outputs
  - ⋆ A description of a group $G_q \in \mathcal{G}$ (including the prime group order q), and
  - ⋆ Two random elements $g \neq 1$, h from G_q (alternatively, $\mathfrak{Gen}$ can choose random elements $g \neq 1$, $w \in \mathbb{Z}_q$ and then set $h = g^w$).

  Elements from G_q are represented with k bits.

- $\mathfrak{Gen}$ is *invulnerable* if it is infeasible, given just a string v generated according to $\mathfrak{Gen}$, to compute a witness w.

Discrete Logarithm Problem, Example

- If G_q is a subgroup of order q in $\mathbb{Z}_p^*$, then the description of G_q consists of two primes p and q. Usually, $|p| > 600$ and $|q| > 160$.

- Group family is the whole sequence $\mathbb{Z}_p^*$ of groups, with G_q being a subgroup of relevant size. The bitlength $|q|$ of q is the security parameter k

- Thus, “feasible” algorithms work in time that is polynomial in k

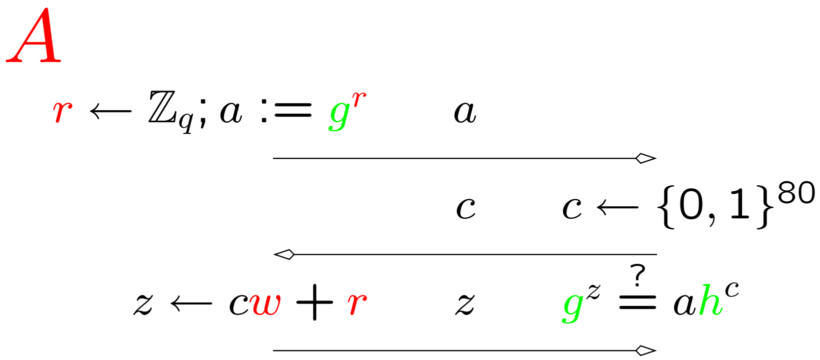
- An invulnerable generator outputs a generator g of large subgroup G_q in some group $\mathbb{Z}_p^*$, s.t. $|q| = k$

Schnorr Identification Scheme (1/2)

Let $\mathcal{G}$ be a family of groups. Let $(G_q, g, w) \leftarrow \mathfrak{Gen}(1^k)$ and let $h := g^w$. Let $v = (G_q, g, h)$ be the common input, w is the private input to A. The corresponding (unique) witness is $w \in \mathbb{Z}_q$ such that $g^w = h$. The relation R consists of all such pairs. , $R = (g^w, w)$.

Schnorr Identification Scheme (2/2)

Let $\mathcal{G}$ be a family of groups. Let $(G_q, g, w) \leftarrow \mathfrak{Gen}(1^k)$ and let $h := g^w$. Let $v = (G_q, g, h)$ be the common input, w is the private input to A. The corresponding (unique) witness is $w \in \mathbb{Z}_q$ such that $g^w = h$. The relation R consists of all such pairs. , $R = (g^w, w)$.



$$\begin{array}{ccc} A & & B \\ r \leftarrow \mathbb{Z}_q; a := g^r & \xrightarrow{\quad a \quad} & \\ & \xleftarrow{\quad c \quad} & c \leftarrow \{0,1\}^{80} \\ z \leftarrow cw + r & \xrightarrow{\quad z \quad} & g^z \overset{?}{=} ah^c \end{array}$$

Check: $g^z = g^{cw+r} = g^r(g^w)^c = ah^c$.

Schnorr: Efficiency (1/2)

- Schnorr scheme was originally designed for smartcard applications, both communication and on-line computation are minimized.

- Instead of a, $H(a)$ may be sent in the first step, where H is a hash function with $|H(a)| < |a|$. Then, verification consists of checking that $H(g^z h^{-c}) = H(a)$. There is no known attack (but the brute force) against the case when just $t = 80$ least significant bits were transferred.

Schnorr: Efficiency, (2/2)

- Communication complexity: $\approx t + t + 2t = 4t = 320$ bits.

- On-line signature generation: one $2t \times t$ bit multiplication (and one t-bit addition). Random number generation and exponentiation can be done off-line, during the processor's idle time.

- If the scheme is used only for identification, where the prover has to reply to the challenge in a few seconds, the security parameter t could be lowered, say, to 48 bits.

Security Properties: Special Soundness (1/2)

- Let $v \in \{0,1\}^*$ be a string. A pair of accepting conversations (v, a, c, z) and (v, a, c', z') with $c \neq c'$ is called a *collision*.
  - ⋆ Collision occurs if the same person starts identification two times with the same first message, is answered by a different second message, and is still accepted

- Σ-protocol (A, B) for relation R has the *collision-property* iff the following holds:
  - ⋆ Given a collision for a public key v, there exists an efficient algorithm that on input of a collision for v outputs a witness w such that $(v, w) \in R$.

Special Soundness (2/2)

- Σ-protocol (A, B) for R satisfies *special soundness*, iff it has the collision-property. Thus, the collision-property implies special soundness.

  - $\star$ NB! This only holds under own “simplifying” assumption

- Intuitively, special soundness guarantees that A does not have an incentive to start the same protocol twice with the same message. She must really include some randomness.

Another major concept: Zero-Knowledge (shortly)

- Suppose A and B engage in an execution of their protocol on common input v.

- B wants to verify that A holds a witness w (a proof of a theorem, a secret key, $\ldots$).

- *Zero-knowledge* means roughly that no matter how B behaves as a verifier, he will not learn any information that it could not have computed itself, even before the start of the protocol

- ZK is usually proven by simulating A. (More in a later lecture)

Zero-Knowledge: Limitations

- ZK protocols require more than three moves unless the underlying language is trivial (in $\mathbf{BPP}$). Thus, in principle, none of the three-move protocols handled here can be ZK. Four-move ZK protocols exist (Bellare, Micali and Ostrovsky).

- The very efficient procedure for turning identification schemes into signature schemes, presented later, cannot be used if the identification scheme is ZK (the simulation used for proving the ZKness can be used to forge the signature). Thus, the BMO protocol cannot be used to construct a signature scheme.

Honest Verifier ZK (more in a later lecture)

(A, B) is *honest verifier zero-knowledge* if it is easy to “simulate” conversations with an honest verifier. If, additionally, the simulator works by taking any uniformly chosen challenge c as input and outputs an accepting conversation where c is the challenge (an accepting conversation (v, a, c, z)), then (A, B) is said to be *special honest verifier zero-knowledge*.

HVZK protocols are useful, since the general ZK protocols are far less efficient. Also, HVZK is sufficient in a wide range of applications. There exist transformation methods for turning certain classes of HVZK protocols into ZK ones.

Witness Hiding

- Let (A, B) be a Σ-protocol for relation R with generator $\mathfrak{Gen}$.

- Let A be given an instance $(v_0, w_0) \in R$ as generated by $\mathfrak{Gen}(1^k)$, and let B^* be an arbitrary PPT machine.

- The protocol (A, B^*) can be executed on common input v_0 as many times as B^* desires. This means that A is given to B^* as a black box.

- However, B^* does not control the random tape of A (i.e., he cannot rewind A).

Witness Hiding, cnt

- *Witness hiding* captures the idea that no matter how maliciously the enemy interrogates an honest prover, it gets at most a negligible advantage when trying to compute any w'_0 in $R_W(v_0)$, compared to the situation before the start of the protocol.

- ZK guarantees that no information whatsoever is revealed in case of any fixed common input v_0

- Witness hiding only guarantees that no *useful* information is given away in the average (otherwise $\mathfrak{Gen}$ would not be invulnerable)

Schnorr scheme: Security

Special Soundness. Given two accepting conversations (v, a, c, z) and (v, a, c', z'), with $c \neq c'$, w is computed as

$$w \leftarrow \frac{z - z'}{c - c'} = \frac{(cw + r) - (c'w + r)}{c - c'} = \frac{(c - c')w}{c - c'} \ .$$

Thus, the Schnorr scheme satisfies special soundness.

Special HVZK. Select $c, z \leftarrow \mathbb{Z}_q$, compute $a \leftarrow g^z \cdot h^{-c}$. Then (v, a, c, z) is an accepting conversation with the correct distribution.

It was however not known if Schnorr's scheme is WH. Very recently, Schnorr's scheme's security against impersonation has been finally proven.

M. Bellare and A. Palacio, "GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active and Concurrent Attacks Authors", CRYPTO 2002 (august 2002)

Okamoto-Schnorr Scheme

Let $\mathcal{G}$ be a family of groups. This time,

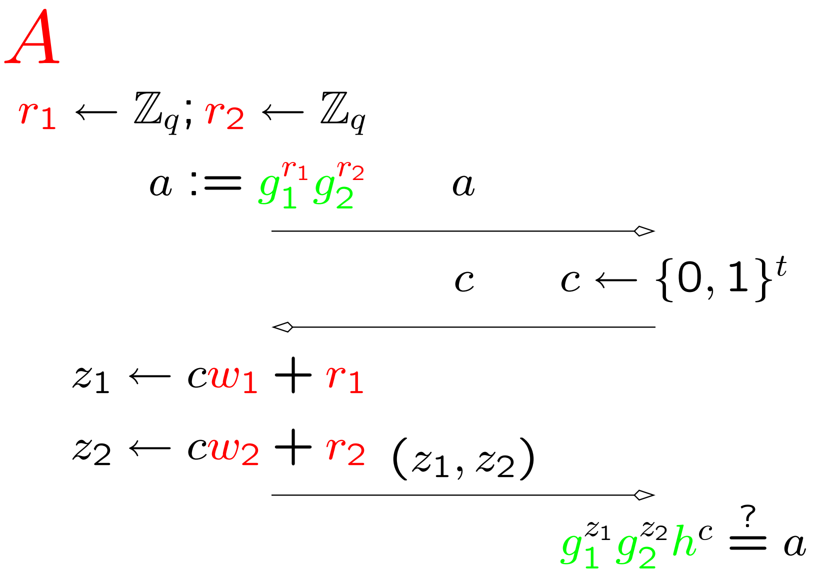
$$(G_q, g_1, g_2, w_1, w_2) \leftarrow \mathfrak{Gen}(1^k) \ ,$$

where both g_1 and g_2 are generators. Let

$$h := g_1^{-w_1} g_2^{-w_2} \ .$$

Let $v = (G_q, g_1, g_2, h)$ be the common input, $w = (w_1, w_2)$ is the private input to A.

Okamoto-Schnorr Scheme



A B

$r_1 \leftarrow \mathbb{Z}_q; r_2 \leftarrow \mathbb{Z}_q$

$a := g_1^{r_1} g_2^{r_2}$ $\xrightarrow{a}$

$\xleftarrow{c}$ $c \leftarrow \{0, 1\}^t$

$z_1 \leftarrow cw_1 + r_1$

$z_2 \leftarrow cw_2 + r_2$ $\xrightarrow{(z_1, z_2)}$

$g_1^{z_1} g_2^{z_2} h^c \overset{?}{=} a$

Check: $g_1^{z_1} g_2^{z_2} h^c = g_1^{cw_1+r_1} g_2^{cw_2+r_2} g_1^{-cw_1} g_2^{-cw_2} = g_1^{r_1} g_2^{r_2} = a$.

Okamoto-Schnorr Scheme: Security

Def (A, B) is *secure* if

1. (A, B) succeeds with overwhelming probability.

2. There is no coalition of A^*, B^* with the property that, after a polynomial number of executions of (A, B^*) and relaying a transcript of the communication to A^*, it is possible to execute (A^*, B) with nonnegligible probability of success.

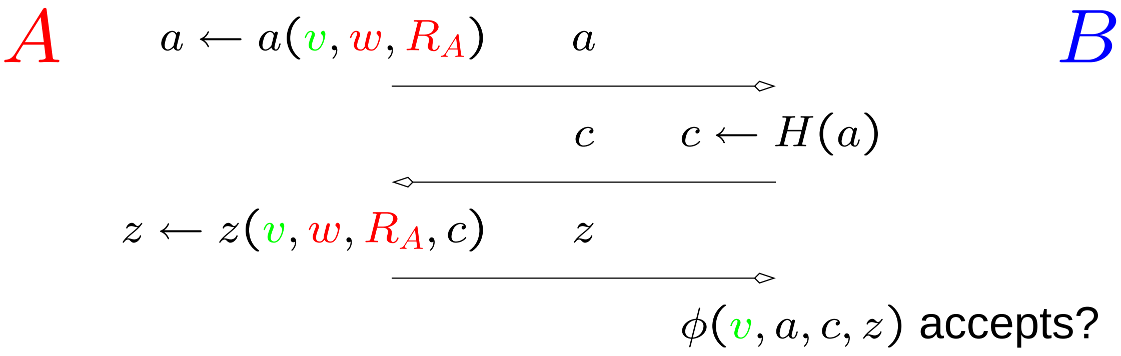
Here, A^* does not know the value w.

Theorem The Okamoto-Schnorr scheme is secure iff the DL is intractable.

Conversion to Signature Scheme

Step I

Σ-protocols can be converted into signature schemes by using the next general method:



A $\quad a \leftarrow a(v, w, R_A)$ $\quad a$ $\quad B$

c $\quad c \leftarrow H(a)$

$z \leftarrow z(v, w, R_A, c)$ $\quad z$

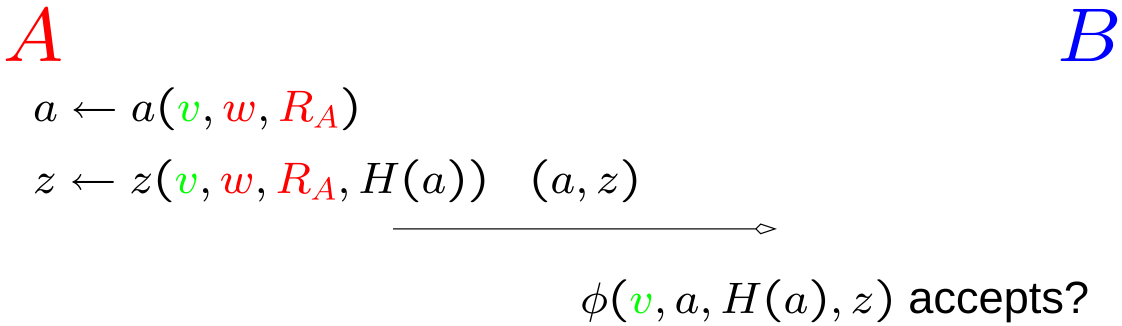
$\phi(v, a, c, z)$ accepts?

Here, H is a random oracle: a function with initially empty database (a, c), such that $H(a)$ returns c if (a, c) is in the database for some c. Otherwise H generates uniformly a new c, adds (a, c) to the database and returns newly generated c.

Conversion to Signature Scheme

Step II

c is a random string that depends provably on the value a (exactly what was needed from the c!). Additionally, A can compute c herself and thus, interaction with B becomes unnecessary.



A $\qquad\qquad\qquad\qquad$ B

$a \leftarrow a(v, w, R_A)$

$z \leftarrow z(v, w, R_A, H(a)) \quad \xrightarrow{(a, z)}$

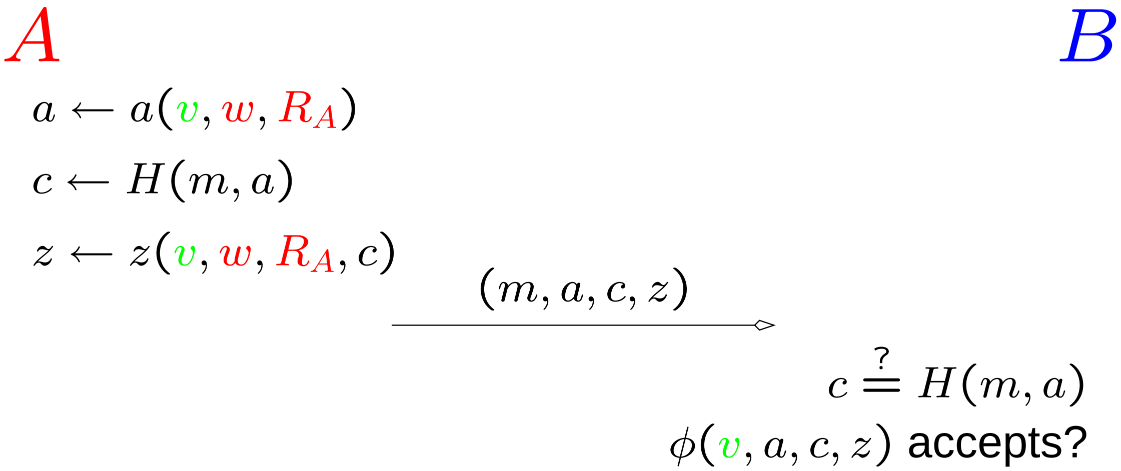
$\phi(v, a, H(a), z)$ accepts?

But: this cannot be used as an identification scheme (A may replay the same a) neither can used for most of the Σ-protocols as a signature scheme (e.g., in the case of Schnorr, a is a function of secret value r).

Conversion to Signature Scheme

Step III

Idea: introduce a message m to be signed:



A $\qquad\qquad\qquad\qquad$ B

$a \leftarrow a(v, w, R_A)$

$c \leftarrow H(m, a)$

$z \leftarrow z(v, w, R_A, c)$

$$\xrightarrow{(m, a, c, z)}$$

$c \overset{?}{=} H(m, a)$

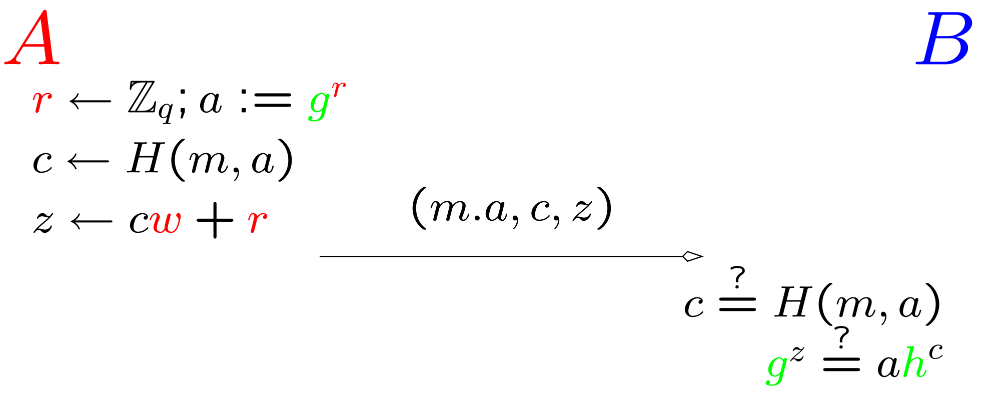
$\phi(v, a, c, z)$ accepts?

Schnorr Signature Scheme

Let $\mathcal{G}$ be a family of groups. Let

$$(G_q, w, h) \leftarrow \mathfrak{Gen}(1^k)$$

and let $h := g^w$. Let $v = (G_q, g, h)$ be the common input, w is the private input to A. The corresponding (unique) witness is $w \in \mathbb{Z}_q$ such that $g^w = h$. The relation R consists of all such pairs.



A

$r \leftarrow \mathbb{Z}_q ; a := g^r$

$c \leftarrow H(m, a)$

$z \leftarrow cw + r$

$\xrightarrow{(m.a, c, z)}$

B

$c \overset{?}{=} H(m, a)$

$g^z \overset{?}{=} ah^c$

Check: $g^z = g^{cw+r} = g^r (g^w)^c = g^w h^c = ah^c$.

SSS: Efficiency

- A has to perform on-line one H evaluation, one 160-bit multiplication and one addition.

- Communication can be reduced: A sends (m, c, z) and B verifies that $s = H(m, g^z h^{-c})$.

Caveats

H can be chosen to be a standard hash function, but in such case the conversion scheme looses provable security (cf the original paper of Schnorr).

For some concrete identification schemes, the conversion works if H is the random oracle, but not for *any* instantiation of H by a real hash function. (Goldwasser, Tauman, 2003)

If both identification scheme and signature are used in the same smartcard, some care has to be taken. Namely, during the identification scheme B can output as the challenge $c = H(m, a)$ for m chosen by her. After receiving z from A, B will own a legitimate signature (a, c, z) of m.

Solution (Schnorr scheme): A sends the 80 least significant bits of a during the step 1. There is no known attack in this case.

More Applications

Aside from identification and signing, Σ-protocols are also extensively used in the following areas:

- Blind signature/digital cash protocols. For example, the Pointcheval-Stern provably secure blind signatures are based on the Okamoto-Schnorr identification scheme.

- Electronic voting. For example, the Cramer-Gennaro-Schoenmakers secure and optimally efficient election scheme is based on the Schnorr identification scheme.

DSA: Digital Signature Algorithm (Standard)

- DSA — a variation of Schnorr's scheme
- g — a generator of G_q, of order q; G_q is a subgroup of $\mathbb{Z}_p^*$
- Schnorr: Signature $(c, z) = (H(m, g^r \mod q), H(m, g^r \mod q)w + r)$, verify that $c = H(m, g^z h^{-c} \mod q)$
- DSA: Define $a \leftarrow (g^r \mod p) \mod q$, $z = (H(m) + wa)r^{-1} \mod q$. Signature is (a, z)
- Verification: Accept if $(g^{H(m)z^{-1}} h^{az^{-1}} \mod p) \mod q = a$

Deterministic Signature Algorithms (1/2)

- If a signature scheme is constructed from identification scheme, it must have inherent randomness

- But there is *no* reason for a signature scheme to be randomized!

- Recent idea: using efficiently computable bilinear maps $\hat{e}$ (Boneh, Lynn, Shacham, 2001)

- Existence of such is known only in only a few cryptographically interesting groups (supersingular elliptic curves, e.g. — Weil and Tate pairings)

Deterministic Signature Algorithms (2/2)

- Assume $\hat{e}(g^a, h^b) = \hat{e}(g, h)^{ab}$ for any g, h, a, b, and that it is hard to find g^{ab}, given g, g^a, g^b (computational Diffie-Hellman assumption)
- For secret k. w, public k. $v = g^w$ and message m, the signature is m^w
- Verification: Check that $\hat{e}(g, m^w) = \hat{e}(v, m)$.
  Really, $\hat{e}(g, m^w) = \hat{e}(g, m)^w = \hat{e}(g^w, m)$
- Benefit: signature is only one group element $\approx$ 80 bits. Signing (one exponentiation) is fast
- Drawback: computing $\hat{e}$ is $\approx$ 10x slower than computing the exponentiation

Other Signature Algorithms

- ECDSA: As DSA but works on elliptic curve groups
- RSA signature scheme: by itself insecure. Can be made secure by using the PSS conversion scheme
- ESIGN, . . . — many other alternatives