T-79.159 Cryptography and Data Security

Lecture 7: Authentication

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

Recap

- Until now, we talked about *confidentiality*: how to keep data secret
- Two long-known problems with the secret key cryptography:
  - ⋆ Key distribution: Diffie-Hellman and derivatives
  - ⋆ Authentication: topic of today's talk

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
  - ⋆ Semi-classically: biometrics, picture

Cryptographic approach

- Proving that you *are* something almost impossible
  - ⋆ Biometrics is often deceiving
  - ⋆ How to do it by email?
- Proving that you *own* something: OK, but own what?
  - ⋆ Own a book with passwords? This is then proving that you know something (passwords!)
  - ⋆ How to do it by email?
- 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

Proofs of Knowledge: Security Criteria

- Criterion 1 (correctness):

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

  ε is "small"

- Criterion 2 (privacy):

  ⋆ After (possibly many) interactions with a prover, V should not be able to pose as P to the third parties

Identification vs Authentication

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 (*non-transferability*)

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: *non-repudiation*).

Non-repudiation $\neq$ Non-transferability!

- MACs: non-transferability, no non-repudiation

Signatures: shortly

- You must authenticate some data m as coming from you
  - ⋆ Everybody can verify that the data is from you
- Important example: data = legal documents
  - ⋆ Signature must be binding
  - ⋆ You may get sued based on your signature. Several countries have digital signature laws

Signatures: shortly

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

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

- *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 a public key cryptosystem, use its secret key for signing and the public key for verification

“Vanilla” 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 $m =^? s^e \mod n$
- Not secure: $m_1^d \cdot m_2^d = (m_1 m_2)^d$

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 by taking A's role
  - ⋆ Cannot be achieved if A's actions are deterministic
- Thus, an identification protocol must include some randomness from both A and B

Identification protocols: idea (2/2)

- To have mutual randomness, A (resp. B) must send a message that depends on B's (resp. A's) random coins

- 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, both *randomness* and *interactivity* are 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 smart-card to get in
- ATM: identify yourself as a legal customer
- Different websites, e-banking

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.

Notation

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

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

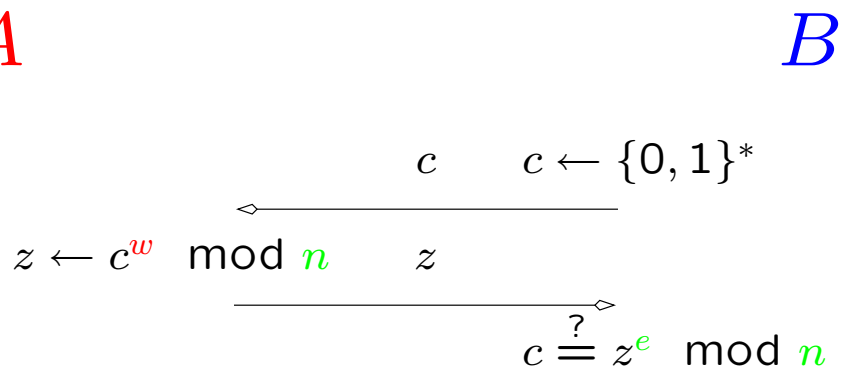
  refers to the computation of the output "a", on input bit string "b".

- For a set V, $v \leftarrow V$ 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 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:

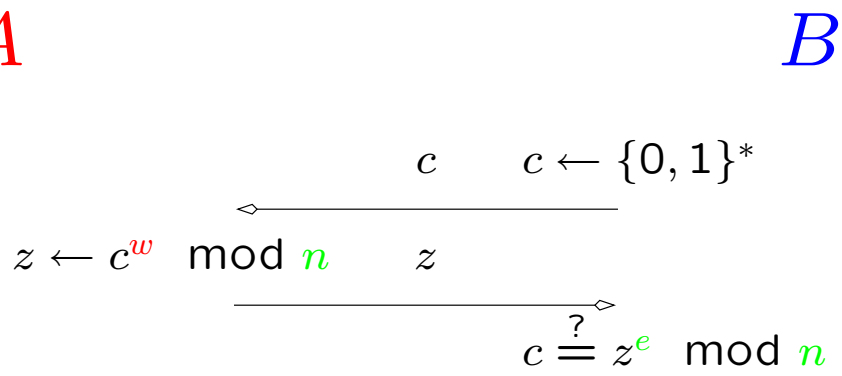


This prevents A from replaying the protocol.

Still bad. Why?

Faulty First Idea for Protocol

- Use 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:



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

Σ-Protocols. General Setting

- 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 modelled as probabilistic polynomial time interactive Turing machines ("efficient algorithms").

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

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

Σ-Protocols. Example

- Secret key is w, public key is $v = g^w$
- There is a relation R between w and v:

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

- We need a Σ-protocol for proving that A knows w, s.t. $R(v, w) = 1$, that is, 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].
  - $\star$ Recall that randomness was necessary

Σ-Protocols. Inputs (2/2)

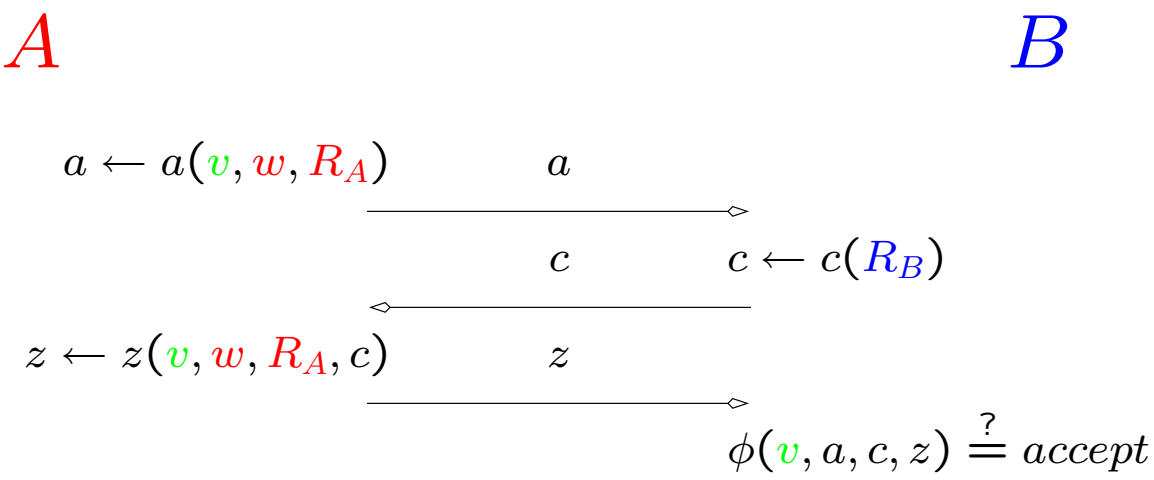
- The pair $(v, w) \in R$, where $R \subset \{0,1\}^* \times \{0,1\}^*$ is a publicly known, typically (but not necessary) efficiently verifiable relation. Let

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

- 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 the Guillou-Quisquater, this is not the case!)

Σ-Protocols. Description



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 $g \in G_q$, $g \neq 1$, then g has order q. 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 w.r.t. g.

- Let $\mathcal{G}$ be a family of groups of prime order such that (a) the group operations can be performed efficiently, (b) group elements can be efficiently sampled with uniform distribution and (c) 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

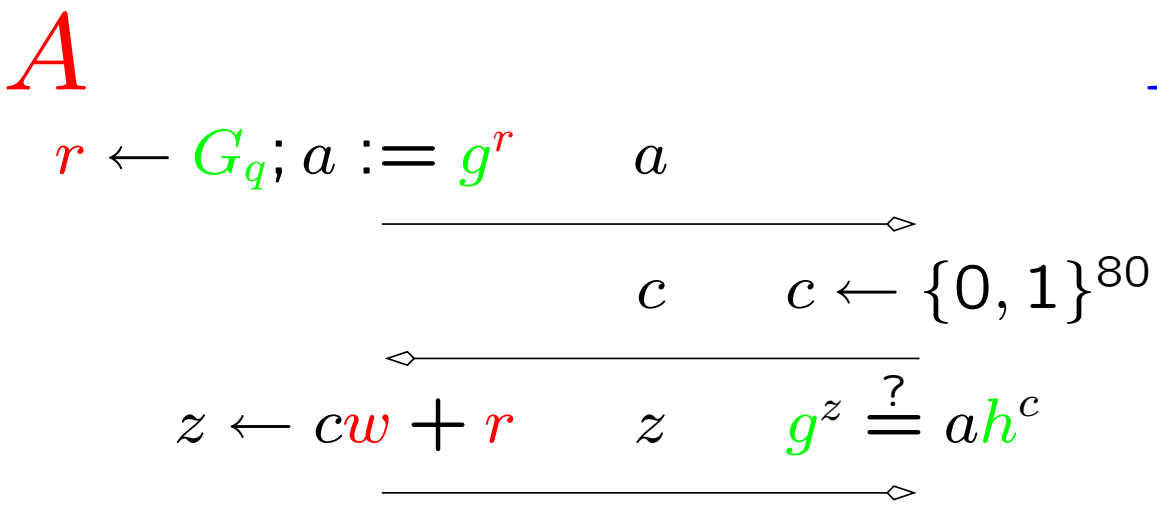
- If G_q is a subgroup of order q in $\mathbb{Z}_p^*$, then the description of G_q consists of two primes p and q. Usually, $|p| > 600$ and $|q| > 160$.
- Group family consists of all groups $\mathbb{Z}_p^*$, with G_q being a subgroup of "relevant" size. The bit-length $|q|$ of q is the security parameter k
- "Feasible" algorithms work in time that is polynomial in k
- An invulnerable generator outputs a generator g of a large subgroup G_q of order 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 be 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.



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

Schnorr: Efficiency

- Schnorr's scheme was originally designed for smart-card applications, both communication and on-line computation are minimised.
- Communication complexity: $\approx |p| + t + |q|$.
- On-line: one $|q| \times 80$ 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 can 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*.

  $\star$ Collision occurs if the same person starts identification two times with the same first message, is answered by a different second message, and is accepted both times

- Σ-protocol (A, B) has *the special soundness* property if the following holds:

  $\star$ 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 $(v, w) \in R$.

(Given security definitions are "simplified")

Special Soundness (2/2)

- Intuitively, special soundness guarantees that A does not have an incentive to start the same protocol twice with the same message.

- She must include some randomness to not reveal her secret.

Another major concept: Zero-Knowledge (shortly)

- A and B execute some protocol on common input v.
- B wants to verify that A holds a witness w (a proof of a theorem, a secret key, . . .).
- *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.

- 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 ZK-ness can be used to forge the signature). Thus, a real ZK protocol cannot be used to construct a signature scheme.

Honest Verifier ZK

- A party is honest/nonmalicious/curious-but-honest when he follows the protocol (though tries to deduce new information from it)
- (A, B) is *honest verifier zero-knowledge* if it is ZK given that B is honest.
- 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 any Σ-protocol for some relation R
- *Witness hiding*: 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
- Difference: Witness hiding only guarantees that no *useful* information is given away in the average

Schnorr scheme: Special Soundness

- Given two accepting conversations (v, a, c, z) and (v, a, c', z')

  $\star$ $g^z = ah^c$ and $g^{z'} = ah^{c'}$

  with $c \neq c'$, w is computed as

  $$w \leftarrow \frac{z - z'}{c - c'} \ ,$$

  since

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

- Thus, the Schnorr scheme satisfies special soundness.

Schnorr scheme: “Special” HVZK

- B must be able to generate an accepting conversation without communicating with Alice

  ⋆ With the same distribution as “real” conversations

- 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 not known if Schnorr’s scheme is witness hiding. Very recently, Schnorr’s scheme’s security against impersonation has been finally proven.

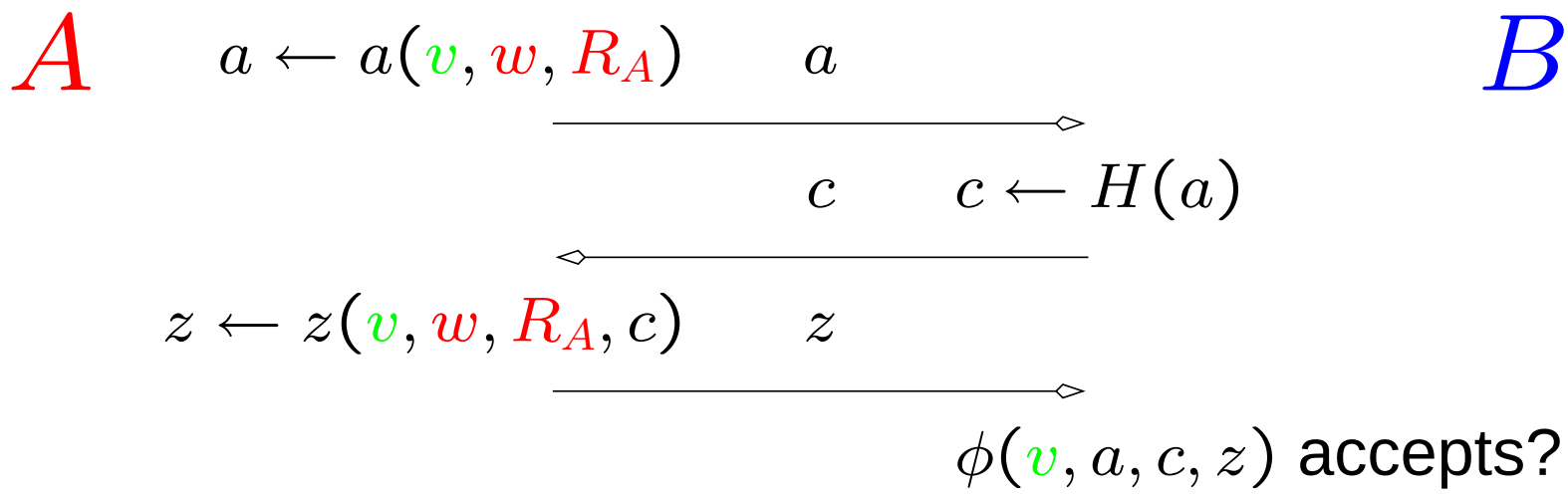
  M. Bellare and A. Palacio, “GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active and Concurrent Attacks Authors”, CRYPTO 2002 (august 2002)

Concept: random oracle

- Random oracle H = random function
  - ⋆ For every x, $H(x)$ is randomly drawn from the output domain
- Implementation:
  - ⋆ H is a subroutine with initially empty database (a, c). $H(a)$ 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.
- In practice, a secure hash function (SHA1) is used

Conversion: Σ-protocol to Signature Scheme

Step I: Assume H is a random oracle. Σ-protocols can be converted into signature schemes by using the next general method:

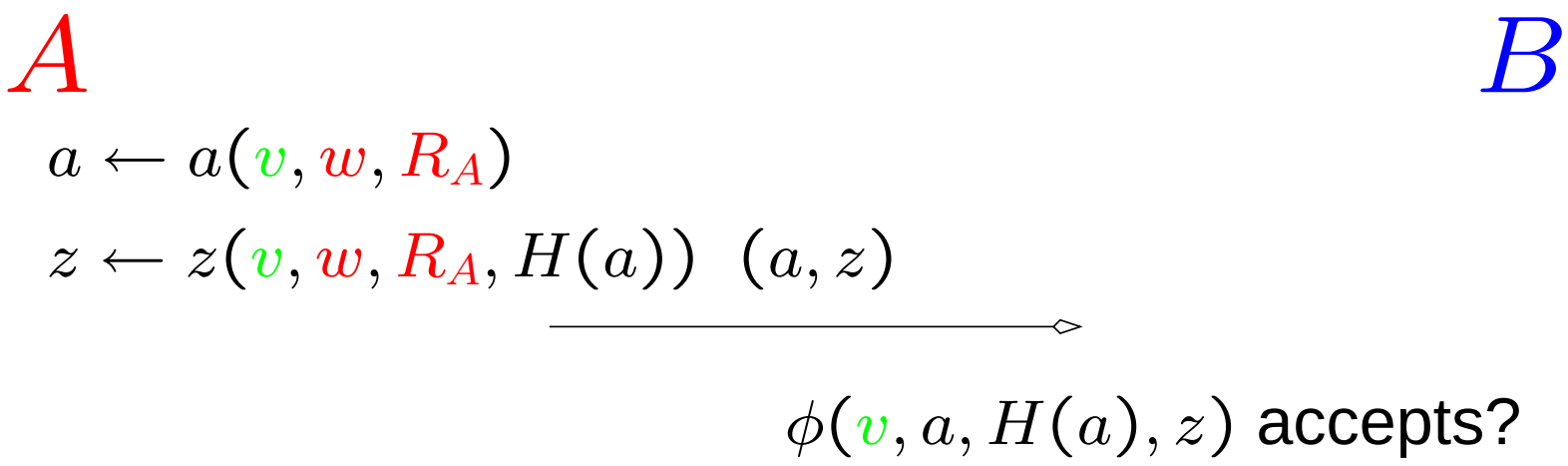


Signature: $(a, H(a), z)$.

c is a random string that depends provably on the value a (exactly what was needed from the c!).

Conversion to Signature Scheme

Step II: A can compute $c = H(a)$ herself and thus, interaction with B becomes unnecessary!



Conversion to Signature Scheme

Step III: Introduce a message m to be signed:

A B

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

$c \leftarrow H(m, a)$

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

(m, a, c, z) →

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

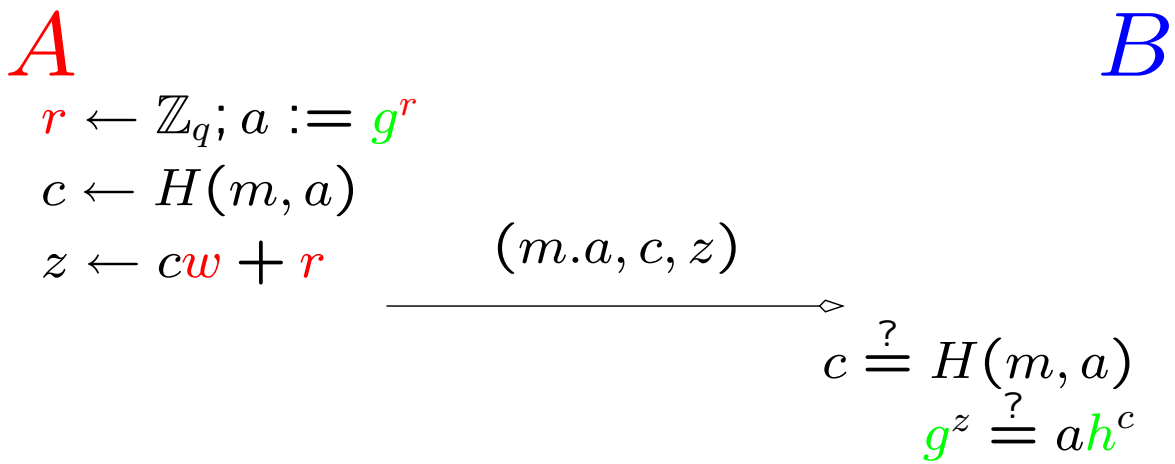
$\phi(v, a, c, z)$ accepts?

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.



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 (1/2)

- H can be chosen to be a standard hash function
- In such case the conversion scheme looses provable security
- 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)

Caveats (2/2)

- If both identification scheme and signature are used in the same smart-card, 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 randomised!
- 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 (super-singular 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 ≈ 80 bits. Signing (one exponentiation) is fast
- Drawback: computing $\hat{e}$ is $\approx$ 10x slower than computing the exponentiation

Gap Diffie-Hellman Assumption (1/3)

- DH problem: given (g, g^a, g^b), compute g^{ab}
- DDH problem: given (g, g^a, g^b, h), decide whether $h = g^{ab}$
- Gap DH assumption in group G: DH is hard but DDH is easy in group G

Gap Diffie-Hellman Assumption (2/3)

- BLS signature scheme: given g, $h = g^x$, m, compute signature as $s = m^x$.

- $m = g^y$ for some y, thus given (g, x, g^y) compute g^{xy}

  ⋆ Forging signature: given (g, g^x, g^y), compute g^{xy} — DH must be hard

- Verification: given (g, g^x, g^y, s), verify $s = g^{xy}$???

  ⋆ Decisional DH must be easy!

- Thus, Gap DH assumption!

Gap Diffie-Hellman Assumption (2/3)

- BLS signature scheme — DH hard, DDH easy
- ElGamal, DH key exchange — DDH hard
- No controversy, just use different groups!

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