Radboud University

Schnorr authentication protocol and signatures

Cryptography, Spring 2020

L. Batina, J. Daemen

May 6, 2020

Institute for Computing and Information Sciences
Radboud University

Outline

Public key authentication protocols

Initial attempts

Schnorr authentication protocol

Schnorr signatures

Conclusions

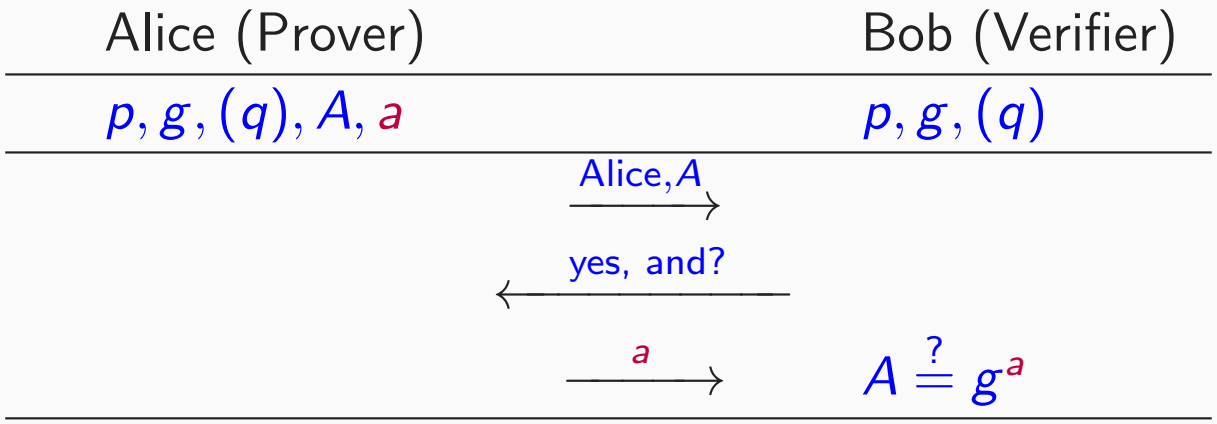
Public key authentication protocols

Public key authentication protocols

- Let's say Bob wants to authenticate Alice
- Password-based authentication
  - over open channel: Eve can learn the password
  - password (or its hash) can leak from Bob's device
- Challenge-response authentication with symmetric crypto
  - Eve cannot learn the secret key
  - secret key can still leak from Bob's device
- What we would like
  - Eve cannot learn the secret
  - secret cannot leak from Bob's device (verifier)
- Solution: challenge-response with public key crypto
  - Alice proves knowledge of private key without revealing it
  - Bob can verify this with Alice's public key only

Initial attempts

Entity authentication: First attempt



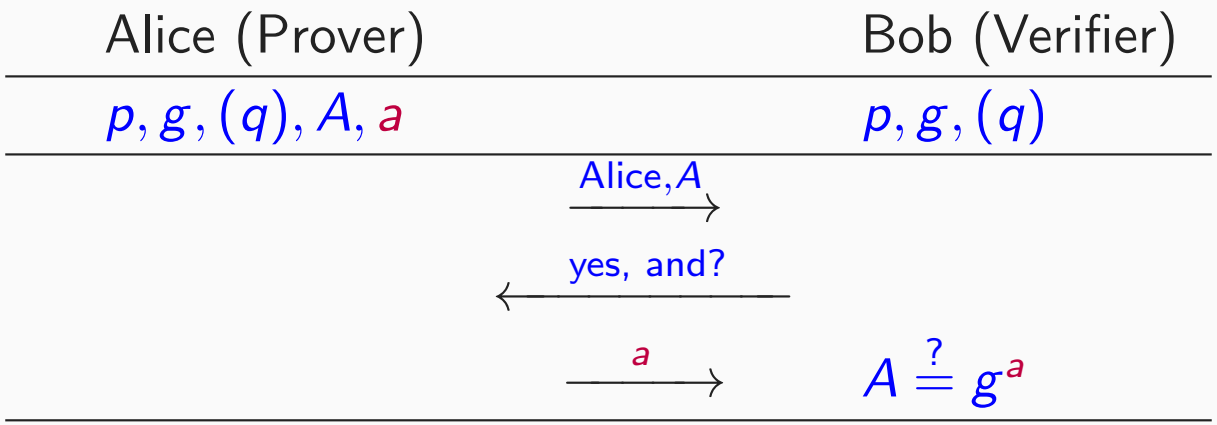
$A \stackrel{?}{=} B$ means: if false the protocol aborts and is said to *fail*.

If a protocol reaches the end, it is said *to succeed*.

Clearly, only Alice can provide a

- Problem: works only once as Alice gave away her private key
  - to Bob
  - but also to Eve if the channel does not provide confidentiality
- Solution: Alice generates a fresh keypair for every run of the protocol
- Not sustainable: Bob must verify Alice's public key for every run

Entity authentication: First attempt (cont'd)



The fact that only Alice can provide a is related to the desirable protocol properties of *completeness* and *soundness*

Completeness

If the prover knows the secret, and prover and verifier run the protocol as specified, the protocol succeeds

In this case Bob gets convinced Alice knows a that satisfies $g^a = A$

Soundness

If the prover does not know the secret, the protocol will only succeed with small probability

Entity authentication: Second attempt

Alice (Prover)		Bob (Verifier)
p, g, q, A, a		p, g, q (Alice: A)
	$\xrightarrow{\text{Hi, Alice here}}$	
	$\xleftarrow{\text{yes, and?}}$	
$v \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$		
$V \leftarrow g^v$	$\xrightarrow{V}$	
$s \leftarrow a + v$	$\xrightarrow{s}$	$A \times V \stackrel{?}{=} g^s$

- Alice sends V, we say: Alice *commits to* V
- Alice sends $s \leftarrow a + v$ to Bob
- Bob accepts if $A \times V = g^s$
- Improvement: s reveals nothing about a as v is random
- It is complete, looks sustainable, but is it sound?

This protocol is not sound

Eve (cheating prover)		Bob (Verifier)
p, g, q, A		p, g, q (Alice: A)
	$\xrightarrow{\text{Hi, Alice here}}$	
	$\xleftarrow{\text{yes, and?}}$	
$s \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$		
$V' \leftarrow g^s$		
$V \leftarrow A^{-1} \times V'$	$\xrightarrow{V}$	
	$\xrightarrow{s}$	$A \times V \stackrel{?}{=} g^s$

Eve just successfully executed the protocol!

- Trick: computing first s and from that V
- Eve does not know the private key corresponding to V
- Eve does not know or learn Alice's private key a

How can we prevent Eve from cheating?

- Bob knows A and gets V
- The prover sends s that satisfies $g^s = A \times V$
- Difference between the honest prover Alice and imposter Eve
  - Alice knows both a and v
  - Eve knows neither a nor v
- Bob could additionally ask for the ephemeral private key v
  - but then Bob would learn a
  - as well as anybody listening in on the line
- Solution: Bob challenges prover by asking for either s or v
  - checks $g^s = A \times V$ or $g^v = V$
  - honest prover Alice can always give the right response
  - Eve can answer v if she followed the protocol, but not s
  - Eve can answer s if she did not follow the protocol, but not v
  - this is the Chaum-Evertse-van de Graaf protocol

Chaum-Evertse-van de Graaf (CEG) protocol

A so-called *zero-knowledge proof of knowledge*: Alice proves she knows a to Bob who has (Alice: A), and reveals nothing in the process

	Alice		Bob	
	p, g, q, A, a		p, g, q (Alice: A)	
	$v \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$			
	$V \leftarrow g^v$	$\xrightarrow{\text{Alice}, V}$	$c \xleftarrow{\$} \{0, 1\}$	
		$\xleftarrow{c}$		
if $(c = 0)$	$r \leftarrow v$			
else	$r \leftarrow v - a$	$\xrightarrow{r}$	if $(c = 0)$	$V \stackrel{?}{=} g^r$
			else	$V \stackrel{?}{=} g^r A$

Three passes: commitment V, challenge c, response r

David Chaum, Jan-Hendrik Evertse en Jeroen van de Graaf, 1987

Can Eve still cheat?

Eve cheating in the CEG protocol

Eve anticipates that challenge will be 0. On a good day:

Eve		Bob
p, g, q, A		p, g, q, (Alice: A)
$r \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$, $V \leftarrow g^r$	$\xrightarrow{\text{Alice}, V}$	$c \xleftarrow{\$} \{0, 1\}$
	$\xleftarrow{c(=0)}$	
	$\xrightarrow{r}$	$V \stackrel{!}{=} g^r$

On a bad day:

Eve		Bob
$r \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$, $V \leftarrow g^r$	$\xrightarrow{\text{Alice}, V}$	$c \xleftarrow{\$} \{0, 1\}$
	$\xleftarrow{c(=1)}$	
I'm outta here!		

Eve cheating in the CEG protocol

Now Eve anticipates that challenge will be 1. On a bad day:

Eve		Bob
p, g, q, A		p, g, q (Alice: A)
$r \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$, $V \leftarrow g^r A$	$\xrightarrow{\text{Alice}, V}$	$c \xleftarrow{\$} \{0, 1\}$
	$\xleftarrow{c(=0)}$	
I'm outta here!		

On a good day:

Eve		Bob
$r \xleftarrow{\$} \mathbb{Z}/q$, $V \leftarrow g^r A$	$\xrightarrow{\text{Alice}, V}$	$c \xleftarrow{\$} \{0, 1\}$
	$\xleftarrow{c(=1)}$	
	$\xrightarrow{r}$	$V \overset{!}{=} g^r A$

If Eve guesses challenge will be 0, she just follows the protocol, otherwise she cheats. If and only if she made the right guess, the protocol succeeds. So her success probability is $1/2$

Iterating the CEG protocol

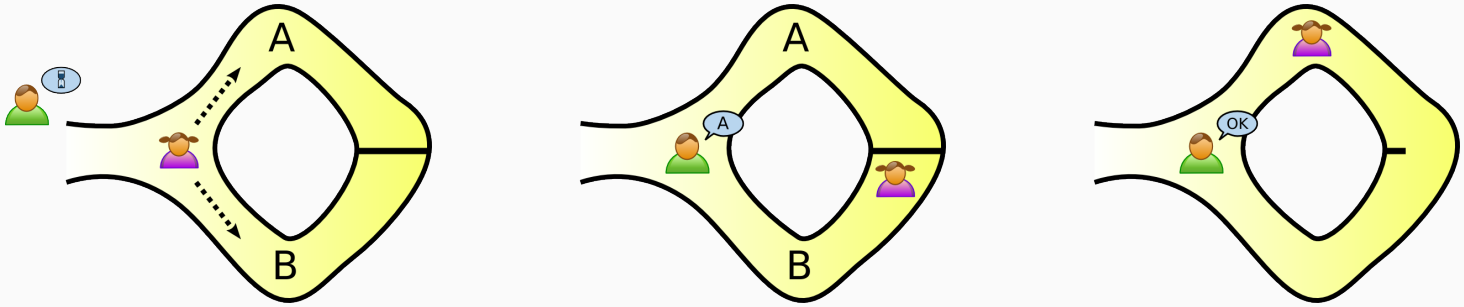
	Alice		Bob	
	p, g, q, A, a		p, g, q (Alice: A)	
.				
For i from 1 to n:				
	$v_i \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$			
	$V_i \leftarrow g^{v_i}$	$\xrightarrow{\text{Alice}, V_i}$	$c_i \xleftarrow{\$} \{0,1\}$	
		$\xleftarrow{c_i}$		
if $(c_i = 0)$	$r_i \leftarrow v_i$			
else	$r_i \leftarrow v_i - a$	$\xrightarrow{r_i}$	if $(c_i = 0)$	$V_i \stackrel{?}{=} g^{r_i}$
			else	$V_i \stackrel{?}{=} g^{r_i} A$

Note: these n iterations can be done in parallel, so with only 3 messages

Eve's success probability is now shrinks to 2^{-n}

CEG explained using Ali Baba's cave

Cave with 2 tunnels connected by a door opening with magic word



Alice (in red) convinces Bob (in green) she knows the magic word:

(1) Bob asks Alice to pick a side while he does not watch (commitment)
(2) Bob asks Alice to come out from a side of his choice (challenge)
(3) Bob witnesses Alice coming out of that side (response)

If Alice repeatedly succeeds, it must be that she knows the magic word

This is complete, sound and *zero-knowledge*: Bob learns nothing from the protocol runs except that Alice knows the magic word

Discussion on zero-knowledge proofs

Zero-knowledge: Bob (or Eve) learns nothing from the protocol except that what is proven

This is formalized by the notions of *transcript* and *simulator*

Definition of transcript

A transcript of a protocol is the sequence of messages exchanged

In case of CEG (Alice, V, c, r): the commitment, challenge and response

Definition of simulator

An algorithm that generates transcripts

Definition of zero-knowledge

A protocol is zero-knowledge if there exists an efficient simulator that given only public information can generate transcripts that cannot be distinguished from transcripts of valid protocol runs

Schnorr authentication protocol

Towards the Schnorr authentication protocol

- The iterated CEG authentication protocol is very costly:
  - computation: both Alice and Bob do s exponentiations
  - communication: n elements of $\langle g \rangle$, n elements of $\mathbb{Z}/q\mathbb{Z}$ and n challenges

Let's rewrite the protocol:

Alice		Bob
p, g, q, A, a		p, g, q (Alice: A)
$v \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$		
$V \leftarrow g^v$	$\xrightarrow{\text{Alice}, V}$	$c \xleftarrow{\$} \{0,1\}$
	$\xleftarrow{c}$	
$r \leftarrow v - ca$	$\xrightarrow{r}$	$V \stackrel{?}{=} g^r A^c$

How can we generalize this? Instead of $c \xleftarrow{\$} \{0,1\}$ do $c \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$

Schnorr authentication protocol

Alice		Bob
p, g, q, A, a		p, g, q (Alice: A)
$v \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$		
$V \leftarrow g^v$	$\xrightarrow{\text{Alice}, V}$	$c \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$
	$\xleftarrow{c}$	
$r \leftarrow v - ca$	$\xrightarrow{r}$	$V \stackrel{?}{=} g^r A^c$

Note: computation of $v - ca$ is done modulo q

- Alice proves knowledge of her private key a
- Bob only needs to have authenticated Alice's public key
- Eve can still cheat by guessing c but success probability is $1/q$
- **Attention**: v shall be chosen randomly and freshly for every protocol run and **never** leak

Claus Schnorr 1991

On the security of the Schnorr protocol

- Completeness:
  - Alice can always deliver response r such that protocol succeeds
  - Schnorr has absolute and unconditional completeness
- Soundness
  - imposter succeeds if guessing c before committing to V
  - She can choose $r \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$ and then compute $V \leftarrow g^r A^c$
  - Success probability of guessing correctly is $1/q$
  - Other attack route: computing a from A: discrete log
  - Schnorr is sound on the condition that DL is hard
- Zero-knowledge:
  - transcript: (Alice, V, c, r)
  - simulator does $c, r \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$ and then $V \leftarrow g^r A^c$
  - Schnorr is (honest-verifier) zero-knowledge: honest-verifier means that challenges should be generated randomly

Schnorr signatures

Message origin authentication

- Alice wants to send a message m to Bob in such a way that Bob has some proof that it is really coming from Alice
- With symmetric crypto:
  - Alice and Bob have a shared key K
  - Alice computes tag T over m using K: $T \leftarrow \mathrm{MAC}_K(m)$
  - Alice sends m, T to Bob
  - Bob verifies the tag: $T \stackrel{?}{=} \mathrm{MAC}_K(m)$
- We want to do this in a way that Bob needs no secret key K
- This would allow Alice to generate a tag verifiable by anyone
- Also, Bob cannot leak the secret
- We would like to do this based on the Schnorr authentication protocol

Making an interactive protocol non-interactive

- CEG and Schnorr authentication are interactive protocols
  - 3 messages: commit, challenge, response
  - they are examples of so-called Σ-protocols
- We wish to make the protocol non-interactive so that
  - prover can compute the transcripts by herself and just send it to verifier without interaction
  - multiple verifiers can verify the transcript: it becomes a *proof*
- For soundness in CEG and Schnorr it is essential that
  - prover receives the challenge after sending the commitment
  - prover can only guess the challenge with negligible success probability
- Protocol transcript as such proves nothing as anyone can generate it
  - this is a consequence of zero-knowledge property
- Still, making protocol non-interactive while keeping soundness is possible!

The Fiat-Shamir transform (Amos Fiat and Adi Shamir, 1986)

- Basic ideas:
  - the output of a random oracle ($\mathcal{RO}$) is unpredictable
  - a cryptographic hash function should behave like a $\mathcal{RO}$
- Prover generates the challenge c as a hash of the commitment V
  - verifier checks if c is indeed the hash of V
  - also include in hash input her public key (p, g, A)
  - this makes the pair V, c only valid for this particular prover
- So $c \leftarrow \mathrm{H}(p; g; A; V)$
  - $x; y; z$: injective encoding of sequence x, y, z in a string
  - for Schnorr, hash output shall be converted to element of $\mathbb{Z}/q\mathbb{Z}$
- Security:
  - as $\mathcal{RO}$ is unpredictable, prover can't predict c when choosing V
  - cheating requires , for given g and $p; g; A$, finding (V, r) that satisfies $V = g^r A^{\mathrm{H}(p;g;A;V)}$
  - if H behaves like a random oracle and DL is hard, this is hard
- The transcript (Alice, V, c, r) now proves knowledge of a private key

Schnorr signature

- Public-key equivalent of a tag
- called *electronic signature*
  - generation requires private key a
  - for verification public key A suffices
- Schnorr signature: Alice signs m

Alice		Bob
p, g, q, A, a		p, g, q (Alice: A)
$v \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$		
$V \leftarrow g^v$		
$c \leftarrow \mathrm{H}(p; g; A; V; m)$		
$r \leftarrow v - ca$	$\xrightarrow{m,(r,V)}$	$c \leftarrow \mathrm{H}(p; g; A; V; m)$
		$V \stackrel{?}{=} g^r A^c$

This is: Fiat-Shamir transform applied to Schnorr authentication protocol with additional inclusion of message in hash input

Schnorr signature: discussion

Alice		Bob
p, g, q, A, a		p, g, q (Alice: A)
$v \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$, $V \leftarrow G^v$		
$c \leftarrow \mathrm{H}(p; g; A; V; m)$		
$r \leftarrow v - ca$	$\xrightarrow{m,(r,V)}$	$c \leftarrow \mathrm{H}(p; g; A; V; m)$
		$V \stackrel{?}{=} g^r A^c$

- Non-interactive: sign, send and verify
- Signature is the transcript (V, c, r)
- As c can be computed from V, it is actually (V, r)
- Forging a signature is hard if H behaves like a $\mathcal{RO}$ and DL is hard
- **Attention**: v shall be chosen randomly and **never** leak
  - signatures of $m \neq m'$ using same commitment leak private key
  - try this

Digital Signature Algorithm (DSA)

- NIST standard FIPS 186
- Proposed in 1991, became standard in 1993
- Also called DSS: Digital Signature Standard
- Purpose: royalty-free alternative for (patented) RSA signatures
- Schnorr-like but not exactly as Schnorr had patent (expired 2008)
- The thing looks like this [for information only]:

Alice		Bob
p, g, q, A, a		p, g, q (Alice: A)
$v \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$, $V \leftarrow g^v$		
$c \leftarrow V \bmod q$		
$r \leftarrow v^{-1}(\mathrm{H}(m) + ca)$	$\xrightarrow{m,(r,c)}$	$w \leftarrow r^{-1}$
		$c \stackrel{?}{=} \left(g^{\mathrm{H}(m)w} A^{cw}\right) \bmod q$

Conclusions

Conclusions

- CEG and Schnorr protocol allow proving the knowledge of a discrete log private key
- This can be used for entity authentication
- 3 criteria for protocols: completeness, soundness and zero-knowledge
- Both (iterated) CEG and Schnorr are complete and sound
- Zero-knowledge
  - CEG is zero-knowledge but rather heavy for usual security strength levels
  - Schnorr is only *honest-verifier* zero-knowledge but lighter
- Both protocols are essentially interactive proofs
- Fiat and Shamir convert them to non-interactive proofs
- Including the message in challenge computation leads to signatures
- We discussed Schnorr signatures and saw DSA