George Danezis
Microsoft Research, Cambridge, UK

Identity and anonymity protocols

Global outline

- Theme: protecting identity on the Internet
- 4 lectures (4 x 1.5h = 6h)
  - Authentication
    - Authentication (4th Dec, part I)
    - Simple anonymous credentials (4th Dec, part II)
  - Anonymous communications & traffic analysis
    - High latency (6th Dec, part I)
    - Low latency (6th Dec, part II)
- Practical deployment and relation with computer security

Secure Authentication

Forward secrecy, privacy, Denial of Service protection, weak passwords…

Why Authentication?

- Authentication protocols
  - Check the assertion a user makes about her identity
  - Studied very early
    - (e.g. Needham-Schroeder 1978 – already sophisticated)
  - Large volume of literature & Formal analysis
    - (Formal / Dolev-Yao model)
- Why such fuss?
  - Key role in computer security
  - Access control matrix:
    - Describe what operations <u>subjects</u> can perform on objects.
  - Identify subject to make decision!

Authentication first!

- Old days – UNIX, mainframes, …
  - Authentication: first interaction with system.
  - Known users interact with few known systems.
  - `Username` and `password` requested and transmitted in clear – user authentication.
    - Context dedicated lines linking terminals to mainframe!
    - If you were in the terminal room you were already ok.
    - Physical security important and strong.
  - Shared keys used for network authentication between mainframes
    - Too few for key management to be an issue

Authentication last?

- Today – Internet
  - Substantial public space requires no authentication
    - DoS, Phishing, …
  - Business with strangers
    - No pre-existing shared keys
    - Public key cryptography needed!

- Anyone can talk to a network host
  - Authentication is last!
  - Transmitted over the insecure network.
  - Adversaries lurking everywhere!
    - Eavesdropping, Phishing, Denial of Service, credential stealing, …

Our focus

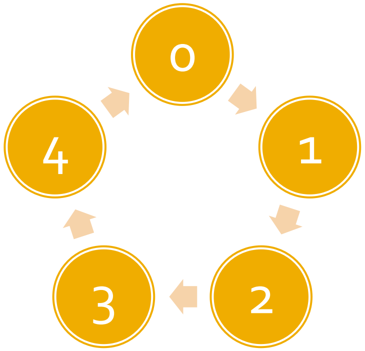
- Study two protocols used for authentication
- Just Fast Keying (JFK)
    - (W. Aiello, S. Bellovin, M. Blaze, R. Canetti, J. Ioannidis, A. Keromytis, O. Reingold – 2003)
  - Core security: public key based key exchange
  - Nice features: Denial of Service prevention, privacy, forward secrecy.
  - Roadmap: Diffie-Hellman exchange, JFK, properties
- Password-Authenticated Key exchange (PAK)
    - (Boyko, MacKenzie, Patel – 2000)
  - Password based key exchange
  - Secure against guessing attacks
  - Roadmap: standard password authentication, PAK, (server strengthening)

Revision

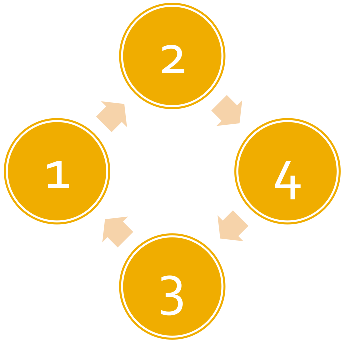
- Discrete logarithm
  and related cryptographic problems
- Diffie-Hellman key exchange
- ISO 9798-3 Authentication protocol

Discrete logarithms (I)

- Assume p a large prime
  - (>1024 bits—2048 bits)
  - Detail: $p = qr+1$ where q also large prime
  - Denote the field of integers modulo p as Z_p
- Example with p=5
  - Addition works fine: $1+2 = 3$, $3+3 = 1$, …
  - Multiplication too: $2*2 = 4$, $2*3 = 1$, …
  - Exponentiation is as expected: $2^2 = 4$



- Choose g in the multiplicative group of Z_p
  - Such that g is a generator
  - Example: g=2



Discrete logarithms (II)

- Exponentiation is computationally easy:
  - Given g and x, easy to compute g^x
- But logarithm is computationally hard:
  - Given g and g^x, difficult to find $x = \log_g g^x$
  - If p is large it is practically impossible
- Related DH problem
  - Given (g, g^x, g^y) difficult to find g^{xy}
  - Stronger assumption than DL problem

Diffie-Hellman (I)

- Alice (A) and Bob (B) do not share any keys
- They want to chat securely
  - Confidentiality (encryption),
  - Integrity (message authentication),
  - Both need a shared key!
- Diffie-Hellman protocol (1976)
  - Key exchange protocol
  - Two parties end up sharing a private key.
- Not authentication yet!

Diffie-Hellman (II)

Public: g, p

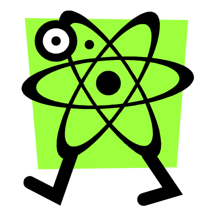
Private: x

Private: y

Alice

$$A \rightarrow B : g^{x} \ (\mathrm{mod}\ p)$$

$$B \rightarrow A : g^{y} \ (\mathrm{mod}\ p)$$



Bob

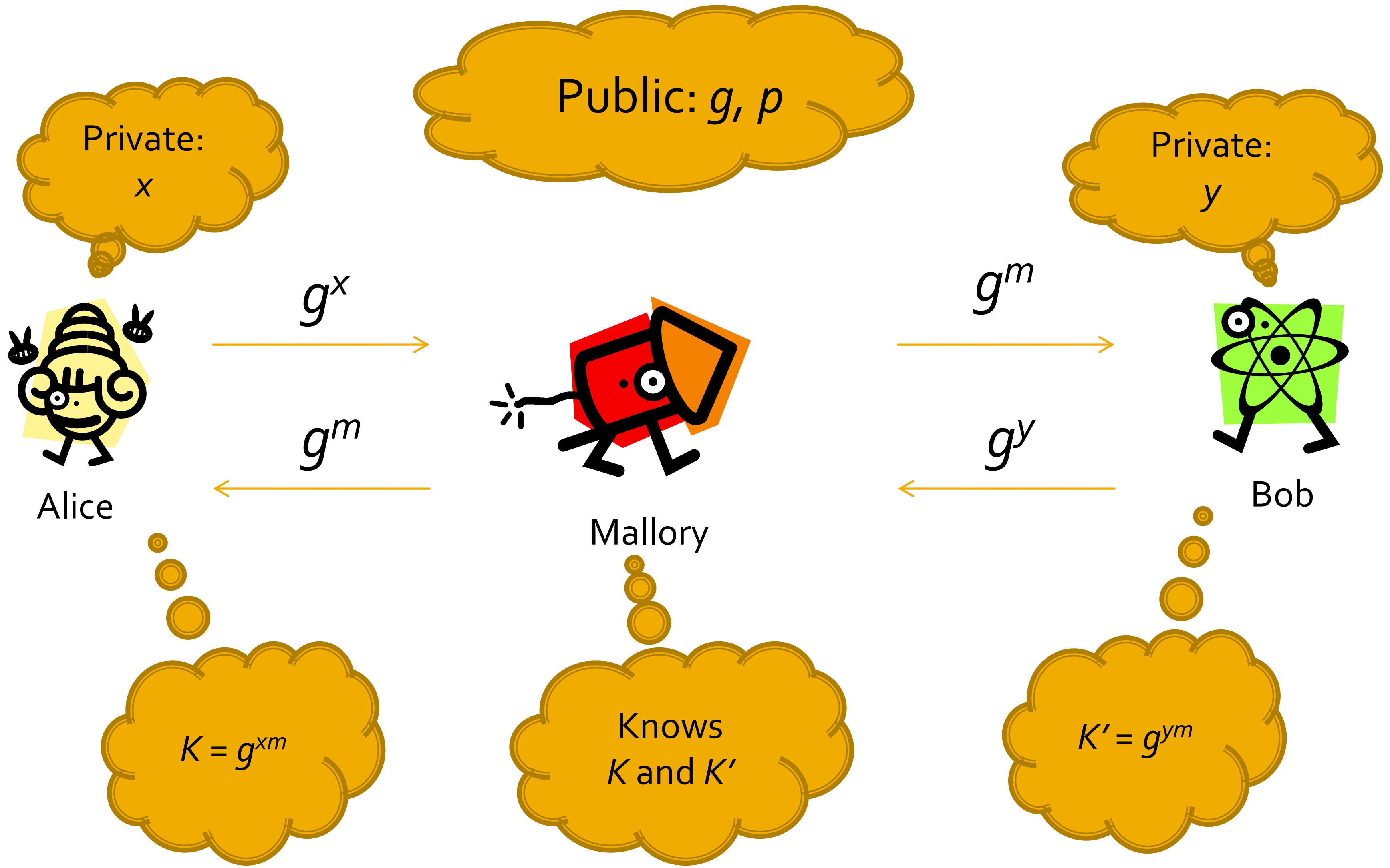
Derive: $(g^y)^x = K = (g^x)^y$

$K = g^{xy}$ is the shared key!

Diffie-Hellman (III)

- Secure against passive adversaries
  - Just looking at messages in the network
  - From *g*, g^x, g^y cannot learn anything about *x*, *y* or g^{xy}
  - Slight problem: *K* is always the same – not fresh!
- Insecure against active adversaries
  - Adversary can delete, insert, modify messages
  - Man-in-the-middle attack

Diffie-Hellman (MITM)



Diffie-Hellman (IV)

- How to secure DH against MITM?
  - Alice and Bob know the fingerprint of each other's keys
    - Telephone directory with hashes of public keys? (Original proposal)
  - PKI: Public Key Infrastructure
    - Trusted party that distributes signed certificates linking names (Alice or Bob, or URLs) to public keys.

- <u>Authenticated key-exchange</u>

ISO 9798-3 – Authenticated DH

- ISO: International Standards Organization
  - (ISO 216 defines the A4 paper size)
- Improves on the Diffie-Hellman exchange:
  - Freshness of keys
    - Both parties contribute fresh random numbers to be used as part of key derivation.
  - MITM protection using long term signature keys
    - Verification keys for the signatures of Alice and Bob are know to each other.
    - (Probably though some PKI)

Some Notation

- $\text{Sign}_A[M]$ – Signature of M with A's key (There is a certificate linking A and her key)
- $H[M]$ – Hash function
- $H_K[M]$ – Keyed hash function
- $\{M\}_K$ – Symmetric Encrypt & MAC

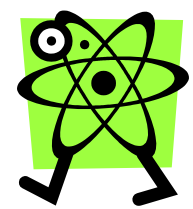
ISO 9798-3 (I)

Private: a

Public: g, p, Ver_A, Ver_B

Private: b

Alice



Bob

B -> A: N_B, B, g^b

A -> B: N_A, N_B, A, g^a, $Sign_A[N_A, N_B, g^a, g^b, B]$

B -> A: N_A , N_B , $Sign_B[N_A, N_B, g^a, g^b, A]$

Random N_A

Random N_B

Derive: $K = H(g^{ab}, N_A, N_B, A, B)$

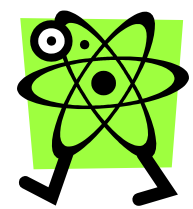
ISO 9798-3 as Diffie-Hellman

Private: a

Public: g, p, Ver_A, Ver_B

Private: b

Alice



Bob

B -> A: N_B, B, g^b

A -> B: N_A, N_B, A, g^a, $Sign_A[N_A, N_B, g^a, g^b, B]$

B -> A: N_A , N_B , $Sign_B[N_A, N_B, g^a, g^b, A]$

Derive: $K = H(g^{ab}, N_A, N_B, A, B)$

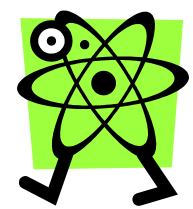
ISO 9798-3– Freshness

Public:
g, p, Ver_A, Ver_B

Private:
a in Zp

Alice

Private:
b in Zp



Bob

B -> A: N_B, B, g^b

A -> B: N_A, N_B, A, g^a, $Sign_A[N_A, N_B, g^a, g^b, B]$

B -> A: N_A , N_B , $Sign_B[N_A, N_B, g^a, g^b, A]$

K will always be different,
Even if g^a, g^b are reused.

Derive: $K = H(g^{ab}, N_A, N_B, A, B)$

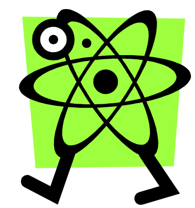
ISO 9798-3– Authenticity

Public:
g, p, Ver_A, Ver_B

Private:
a

Alice

Private:
b



Bob

B -> A: N_B, B, g^b

A -> B: N_A, N_B, A, g^a, $Sign_A[N_A, N_B, g^a, g^b, B]$

B -> A: N_A, N_B, $Sign_B[N_A, N_B, g^a, g^b, A]$

Adversary cannot forge these to do MITM.

Derive: $K = H(g^{ab}, N_A, N_B, A, B)$

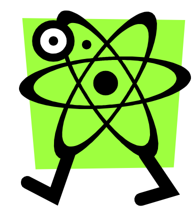
ISO 9798-3– Home work

Private:
a

Public:
g, p, Ver_A, Ver_B

Private:
b

Alice



Bob

B -> A: N_B, B, g^b

A -> B: N_A, N_B, A, g^a, $\text{Sign}_A[N_A, N_B, g^a, g^b, B]$

B -> A: N_A , N_B , $\text{Sign}_B[N_A, N_B, g^a, g^b, A]$

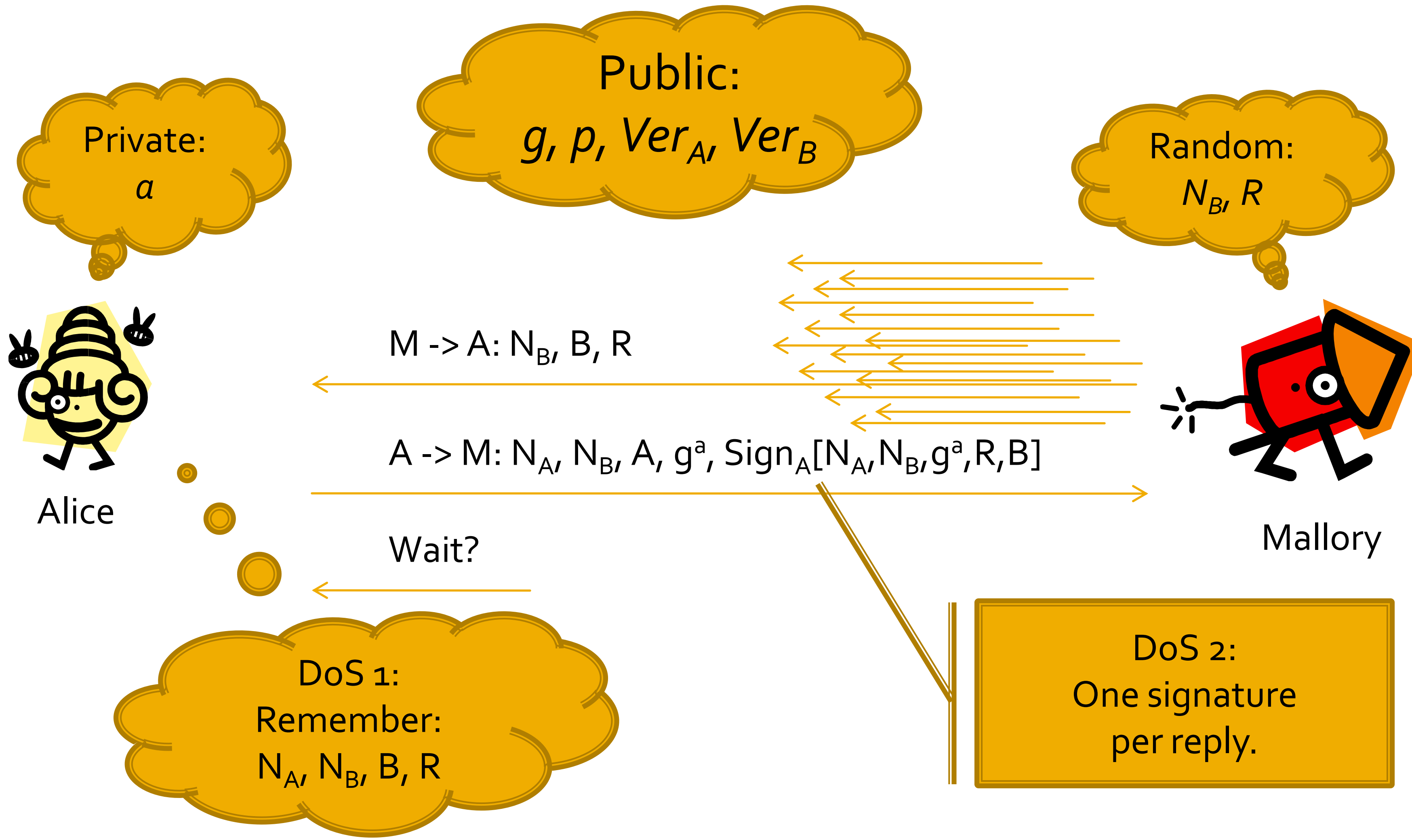
What happens if we do not sign the identities?

Derive: $K = H(g^{ab}, N_A, N_B, A, B)$

Notes on ISO 9798-3

- Forward secrecy – GOOD
  - If g^a and g^b are ephemeral (deleted after the exchange).
  - Revealing the long term signature keys does not compromise K!
- Alice and Bob are certain of each other's identities – GOOD
  - So is any passive eavesdropper
  - Privacy concern – BAD
- Alice maintains state before knowing Bob.
  - Denial of Service: resource depletion (memory) – BAD

ISO 9798-3– Denial of Service



Just Fast Keying (JFKi)

- Authenticated key-exchange
  - All properties of ISO 9798-3
- New properties
  - Denial of Service protection
  - Privacy
    - Initiator's identity is not revealed to third parties.
    - (Responder's identity is revealed.)
- Detailed look at JFKi
  - JFKr – privacy for responder

JFKi (I) – The protocol

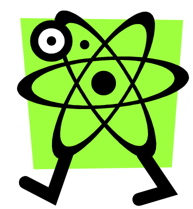
Private: a, HK_A

Public: g, p, Ver_A, Ver_B

Private: b

Alice

(Responder)



Bob

(Initiator)

B -> A: $N'_B, g^b, ID_{A'}$

A -> B: $N'_B, N_A, ID_A, g^a, Sign_A[g^a], H_{HK_A}[g^a, N_A, N'_B, IP_B]$

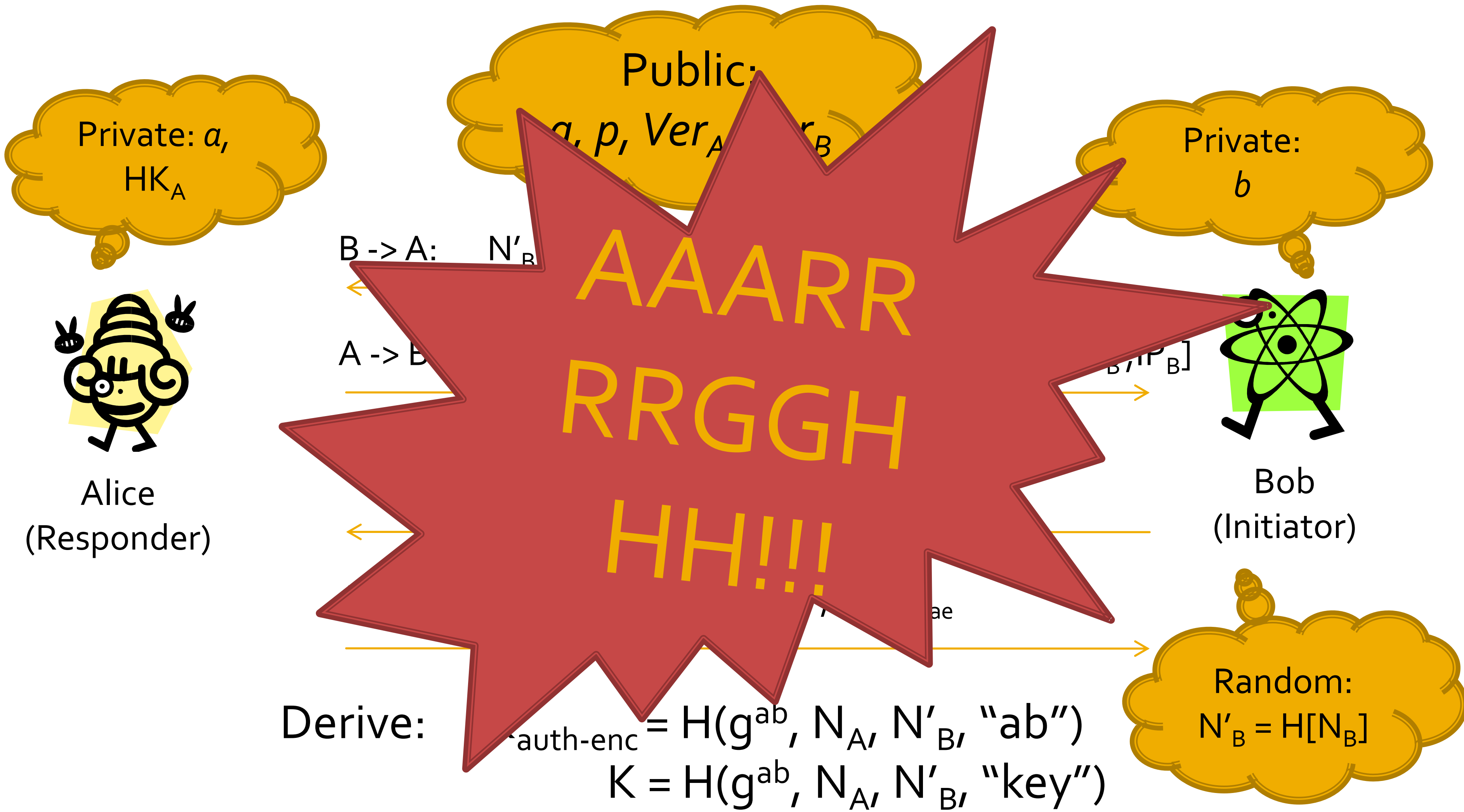
B -> A: $N_B, N_A, g^b, g^a, H_{HK_A}[g^a, N_A, N'_B, IP_B]$
$\{ID_B, Sign_B[N'_B, N_A, g^b, g^a, ID_A]\}_{Kauth\text{-}enc}$

A->B: $\{Sign_A[N'_B, N_A, g^b, g^a, ID_B]\}_{Kauth\text{-}enc}$

Random: $N'_B = H[N_B]$

Derive: $K_{auth\text{-}enc} = H(g^{ab}, N_A, N'_B, \text{“ab”})$
$K = H(g^{ab}, N_A, N'_B, \text{“key”})$

JFKi (II) – The panic



JFKi – The ISO 9798-3 core

Private: a , HK_A

Public: g, p, Ver_A, Ver_B

Private: b

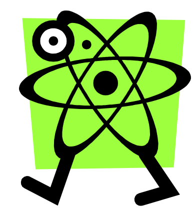
B -> A: N'_B , g^b, $ID_{A'}$

A -> B: N'_B , N_A, ID_A, g^a, $Sign_A[g^a]$, $H_{HK_A}[g^a, N_A, N'_B, IP_B]$

B -> A: N_B , N_A , g^b, g^a, $H_{HK_A}[g^a, N_A, N'_B, IP_B]$
$\{ID_B, Sign_B[N'_B, N_A, g^b, g^a, ID_A]\}_{Kauth\text{-}enc}$

A->B: $\{Sign_A[N'_B, N_A, g^b, g^a, ID_B]\}_{Kauth\text{-}enc}$

Alice
(Responder)



Bob
(Initiator)

Random: $N'_B = H[N_B]$

Derive: $K_{auth\text{-}enc} = H(g^{ab}, N_A, N'_B, \text{“ab”})$

$K = H(g^{ab}, N_A, N'_B, \text{“key”})$

JFKi – Initiator privacy

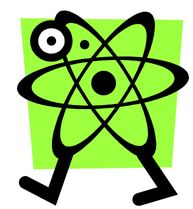
1) Bob can already authenticate Alice

Public:
g, p, Ver_A, Ver_B

Private: a, HK_A

Private: b

Alice
(Responder)



Bob
(Initiator)

B -> A: N'_B, g^b, $ID_{A'}$

A -> B: N'_B, N_A, ID_A, g^a, $Sign_A[g^a]$, $H_{HK_A}[g^a, N_A, N'_B, IP_B]$

B -> A: N_B, N_A, g^b, g^a, $H_{HK_A}[g^a, N_A, N'_B, IP_B]$
$\{ID_B, Sign_B[N'_B, N_A, g^b, g^a, ID_A]\}_{Kauth\text{-}enc}$

A->B: $\{Sign_A[N'_B, N_A, g^b, g^a, ID_B]\}_{Kauth\text{-}enc}$

Random:
$N'_B = H[N_B]$

2) Encryption prevents eavesdropper from learning ID_B

e: $K_{auth\text{-}enc} = H(g^{ab}, N_A, N'_B, \text{“ab”})$
$K = H(g^{ab}, N_A, N'_B, \text{“key”})$

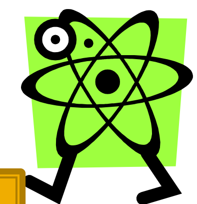
JFKi – DoS Prevention

Public:
g, p, Ver_A, Ver_B

Private: a, HK_A

Private: b

Alice
(Responder)



Bob
(Initiator)

B -> A: $N'_B, g^b, ID_{A'}$

1) Tell ahead of time

A -> B: $N'_B, N_A, ID_A, g^a, Sign_A[g^a], H_{HK_A}[g^a, N_A, N'_B, IP_B]$

Cheap!

B -> A: $N_B, N_A, g^b, g^a, H_{HK_A}[g^a, N_A, N'_B, IP_B]$
$\{ID_B, Sign_B[N'_B, N_A, g^b, g^a, ID_A]\}_{Kauth\text{-}enc}$

A->B: $\{Sign_A[N'_B, N_A, g^b, g^a, ID_B]\}_{Kauth\text{-}enc}$

2) Cookie: Alice uses Bob as secure remote storage!

ive: $K_{auth\text{-}enc} = H(g^{ab}, N_A, N'_B, \text{"ab"})$
$K = H(g^{ab}, N_A, N'_B, \text{"key"})$

Random:
$N'_B = H[N_B]$

Summary of key concepts (1)

USUAL

- Key exchange (DH)
- Authentication of key exchange
  - Freshness
  - Signatures & certificates (PKI)

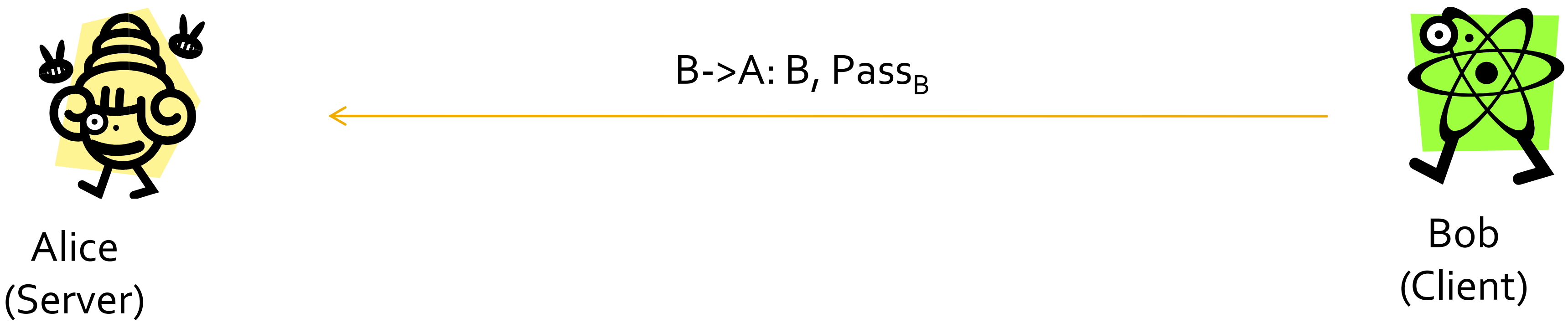
HOT

- Forward secrecy
  - Ephemeral keys
- Privacy
  - Authenticate before telling
  - Protect against passive adversaries
- Denial of service prevention
  - Cookies

What about passwords?

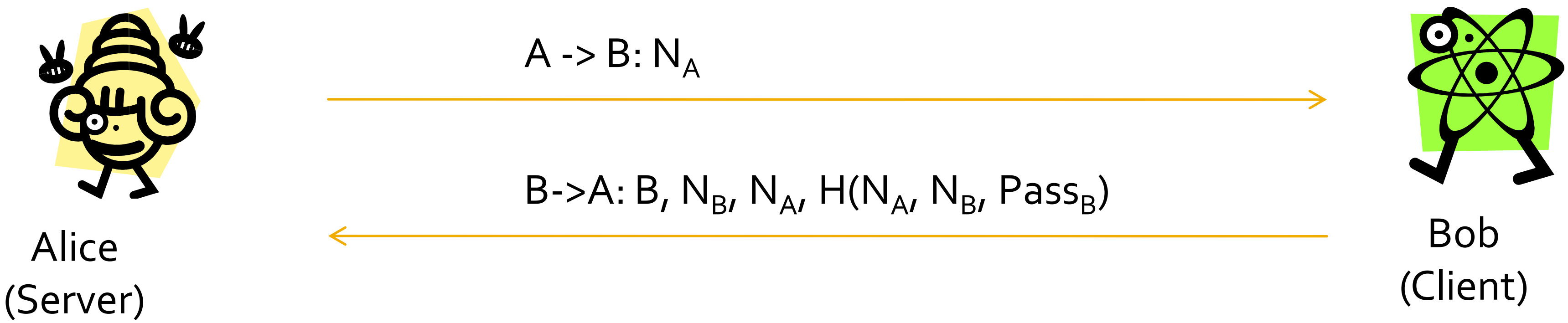
- PKI, certificates, shared cryptographic keys
  - Not very usable
  - Need bootstrapping
- Web authentication
  - Password based
- Small device pairing – using 4-digit PINs
  - Smart phones
  - Bluetooth
  - User interface constraints

Naive password authentication (1)



- Most web services
- Eavesdropper can get $Pass_B$ (SSL?)
- Alice does not store passwords in clear
  - Alice DB: B, S_B, $H(Pass_B, S_B, B)$ – S_B called `salt'
  - Can still check passwords

Naive password authentication (2)



- HTTP digest authentication
- Problem 1: No server authentication
- Problem 2: Off-line guessing attacks
  - Entropy of PIN or passwords small
  - Try all words in dictionary until you get $H(N_A, N_B, Pass_B)$
- Server compromise is bad – no hashing/salting

Password auth. – Requirements

- Alice and Bob share a weak secret
  - a short PIN (4-digits)
  - Password (dictionary word)
  - Low entropy
- Mutual authentication
- Derive a cryptographically strong key
  - Encryption / message authentication
- No off-line guessing attacks
- (Security against server compromise)

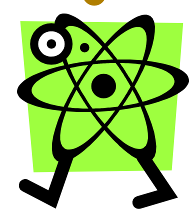
PAK – Definition

Private:
Password: π
Random a

Public:
g, p

Private:
Password: π
Random b

Alice



Bob

A->B: $g^a \cdot (H_1[A,B,\pi]) = m$

B->A: $g^b, H_{2a}[A,B,m,g^b, \underline{(m/(H_1[A,B,\pi]))^b}, \pi] = k$

A->B: $H_{2b}[A,B,m,g^b, \underline{(m/(H_1[A,B,\pi]))^b}, \pi] = k'$

g^{ab}

Check k

Check k′

Derive: $K = H_3[A,B,m, g^b, \underline{(m/(H_1[A,B,\pi]))^b}, \pi]$

PAK – Diffie-Hellman core

Private:
Password: π
Random a

Public:
g, p

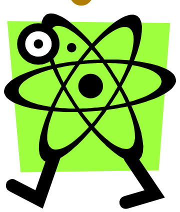
Private:
Password: π
Random b

Alice

A->B: $g^a \cdot (H_1[A,B,\pi]) = m$

B->A: $g^b, H_{2a}[A,B,m,g^b, g^{ab}, \pi] = k$

A->B: $H_{2b}[A,B,m,g^b, g^{ab}, \pi] = k'$



Bob

Derive: $K = H_3[A,B,m, g^b, (m/(H_1[A,B,\pi]))^b, \pi]$

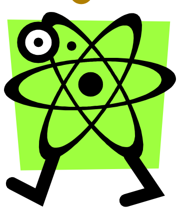
PAK – Authentication?

Private:
Password: π
Random a

Public:
g, p

Private:
Password: π
Random b

Alice



Bob

A->B: $g^a \cdot (H_1[A,B,\pi]) = m$

B->A: $g^b, H_{2a}[A,B,m,g^b,(m/(H_1[A,B,\pi]))^b,\pi] = k$

A->B: $H_{2b}[A,B,m,g^b,(m/(H_1[A,B,\pi]))^b,\pi] = k'$

The ability to blind and un-blind proves knowledge of the password π.

Derive: $K = H_3[A,B,m,g^b,(m/(H_1[A,B,\pi]))^b,\pi]$

PAK – no off-line guessing

Private:
Password: π
Random *a*

Public:
g, *p*

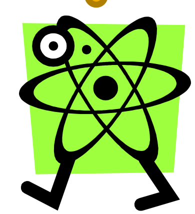
Private:
Password: π
Random *b*

Alice

A->B: $\underline{g^a}\cdot(H_1[A,B,\pi]) = m$

B->A: $g^b, H_{2a}[A,B,m,g^b, \underline{g^{ab}}, \pi] = k$

A->B: $H_{2b}[A,B,m,g^b, \underline{g^{ab}}, \pi] = k'$



Bob

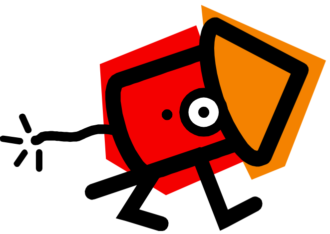
Derive: $K = H_3[A,B,m, g^b, (m/ (H_1[A,B, \pi]))^b, \pi]$

PAK – on-line guessing

Private:
Random x
Random $\underline{\pi_x}$

Public:
g, p

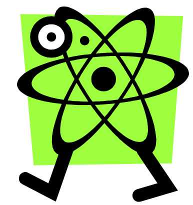
Private:
Password: π
Random b



Mallory

M->B: $\underline{g^x}\cdot(H_1[A,B,\pi_x]) = m$

B->M: $g^b, H_{2a}[A,B,m,g^b,(m/(H_1[A,B,\pi]))^b,\pi] = k$



Bob

Check if
$k \stackrel{?}{=} H_{2a}[A,B,m,g^b,(m/(H_1[A,B,\pi_x]))^b,\pi_x]$
(if $\pi = \pi_x$)

Security precaution: limit the number of attempts!

Practical considerations

- Denial of Service
  - Adversary can lock users out!
  - Require to give up after few attempts.
- Implicit names
  - Alice and Bob expect to talk to each other
  - Otherwise … Privacy concerns
- Public key operation – more expensive than hashing.
- PAK-X: server compromise-resistant.

Summary of key concepts (2)

- A weak password can bootstrap a strong one
  - Force adversary to go active
  - PAK-X modification to allow salting (home work)
- Key problems
  - Denial of service
- Applicability
  - Pairing devices
  - Shy adversaries ☺
  - Not www, login, …

In conclusion

- Expect a lot from your authentication
  - Key derivation (not just identification)
  - Forward secrecy
  - Denial of service prevention
  - Privacy
- More properties
  - Federation, thin client, …
- Do not design your own protocol unless you understand all those in the literature!

References

- Core:
  - William Aiello, Steven M. Bellovin, Matt Blaze, Ran Canetti, John Ioannidis, Angelos D. Keromytis, Omer Reingold: **Just fast keying: Key agreement in a hostile internet**. ACM Trans. Inf. Syst. Secur. 7(2): 242-273 (2004)
  - Victor Boyko, Philip D. MacKenzie, Sarvar Patel: **Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman.** EUROCRYPT 2000: 156-171
- More:
  - Martín Abadi, Bruno Blanchet, Cédric Fournet: **Just fast keying in the pi calculus**. ACM Trans. Inf. Syst. Secur. 10(3): (2007)
  - Colin Boyd, Anish Mathuria: **Protocols for Authentication and Key Establishment**. 2003, XVI, 321 p., Hardcover. ISBN: 978-3-540-43107-7. Springer.

Anonymous credentials

Proving certified attributes without leaking identities

A critique of identity

- Identity as a proxy to check credentials
  - Username decides access in Access Control Matrix
- Sometime it leaks too much information
- Real world examples
  - Tickets allow you to use cinema / train
  - Bars require customers to be older than 18
    - But do you want the barman to know your address?

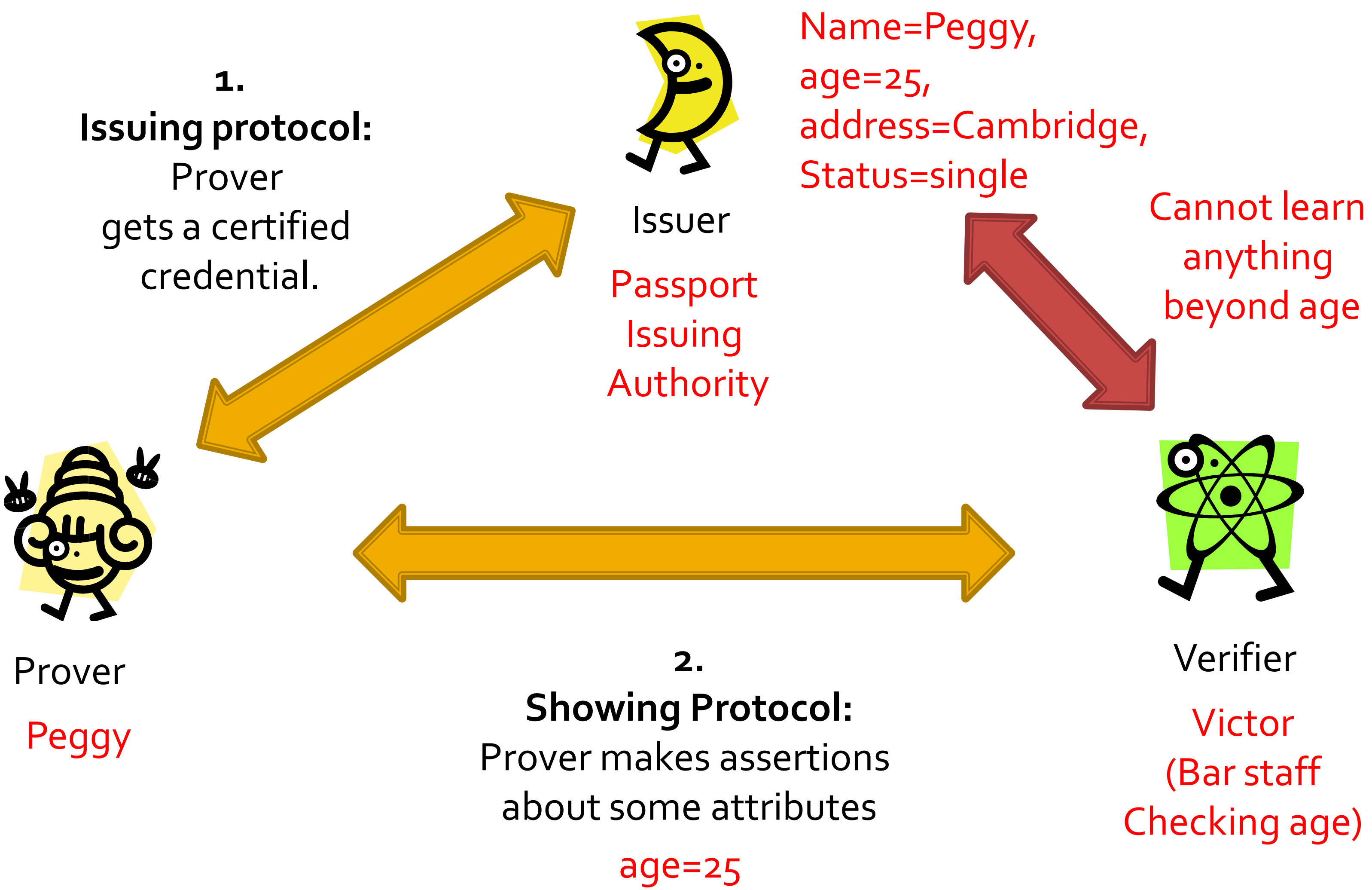
The privacy-invasive way

- Usual way:
  - **Identity provider** certifies attributes of a **subject**.
  - **Identity consumer** checks those attributes
  - Match credential with **live person** (biometric)
- Examples:
  - E-passport: signed attributes, with lightweight access control.
    - Attributes: nationality, names, number, pictures, …
  - Identity Cards: signatures over attributes
    - Attributes: names, date of birth, picture, address, …

Anonymous credentials

- The players:
  - Issuer (I) = Identity provider
  - Prover (P) = subject
  - Verifier (V) = identity consumer
- Properties:
  - The prover convinces the verifier that he holds a credential with attributes that satisfy some boolean formula:
    - Simple example “age=18 AND city=Cambridge”
  - Prover cannot lie
  - Verifier cannot infer anything else aside the formula
  - Anonymity maintained despite collusion of V & I

The big picture



Two flavours of credentials

- Single-show credential (Brands & Chaum)
  - Blind the issuing protocol
  - Show the credential in clear
  - Multiple shows are linkable – BAD
  - Protocols are simpler – GOOD

We will Focus on these

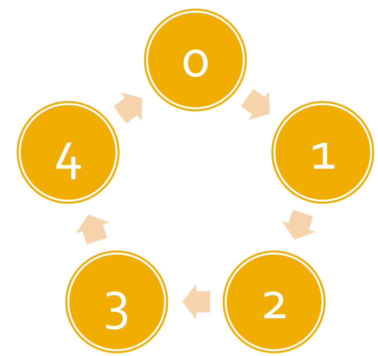
- Multi-show (Camenisch & Lysyanskaya)
  - Random oracle free signatures for issuing (CL)
  - Blinded showing
    - Prover shows that they know a signature over a particular ciphertext.
  - Cannot link multiple shows of the credential
  - More complex – no implementations

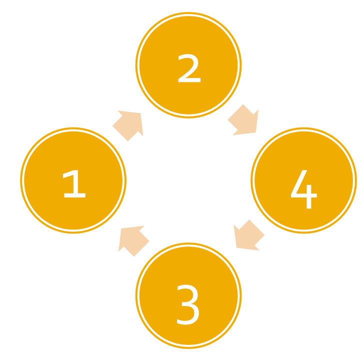
Technical Outline

- Cryptographic preliminaries
  - The discrete logarithm problem
  - Schnorr's Identification protocol
    - Unforgeability, simulator, Fiat-Shamir Heuristic
    - Generalization to representation
- Showing protocol
  - Linear relations of attributes
  - AND-connective
- Issuing protocol
  - Blinded issuing

Discrete logarithms (I) - revision

- Assume p a large prime
  - (>1024 bits—2048 bits)
  - Detail: $p = qr+1$ where q also large prime
  - Denote the field of integers modulo p as Z_p
- Example with $p=5$
  - Addition works fine: $1+2 = 3$, $3+3 = 1$, …
  - Multiplication too: $2*2 = 4$, $2*3 = 1$, …
  - Exponentiation is as expected: $2^2 = 4$
- Choose g in the multiplicative group of Z_p
  - Such that g is a generator
  - Example: g=2





Discrete logarithms (II) -revision

- Exponentiation is computationally easy:
  - Given g and x, easy to compute g^x
- But logarithm is computationally hard:
  - Given g and g^x, difficult to find $x = \log_g g^x$
  - If p is large it is practically impossible
- Related DH problem
  - Given (g, g^x, g^y) difficult to find g^{xy}
  - Stronger assumption than DL problem

More on Z_p

- Efficient to find inverses
  - Given c easy to calculate g^{-c} mod p
    - (p-1) – c mod p-1
- Efficient to find roots
  - Given c easy to find $g^{1/c}$ mod p
    - c (1/c) = 1 mod (p-1)
  - Note the case N=pq (RSA security)
- No need to be scared of this field.

Schnorr's Identification protocol

- Exemplary of the zero-knowledge protocols credentials are based on.

- Players
  - Public – g a generator of Z_p
  - Prover – knows x (secret key)
  - Verifier – knows $y = g^x$ (public key)

- Aim: the prover convinces the verifier that she knows an x such that $g^x = y$
  - Zero-knowledge – verifier does not learn x!

- Why identification?
  - Given a certificate containing y

Schnorr's protocol

Public: g, p

Knows: x

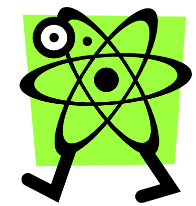
Knows: $y=g^x$

Peggy
(Prover)

P->V: $g^w = a$ (witness)

V->P: c (challenge)

P->V: $cx+w = r$ (response)



Victor
(Verifier)

Random: w

Check:

$g^r = y^c a$

$g^{cx+w} = (g^x)^c g^w$

No Schnorr Forgery (intuition)

- Assume that Peggy (Prover) does not know x?
  - If, for the same witness, Peggy forges two valid responses to two of Victor's challenges

$$r_1 = c_1 x + w$$

$$r_2 = c_2 x + w$$

  - Then Peggy must know x
    - 2 equations, 2 unknowns (x,w) – can find x

Zero-knowledge (intuition)

- The verifier learns nothing new about x.
- How do we go about proving this?
  - Verifier can simulate protocol executions
    - On his own!
    - Without any help from Peggy (Prover)
  - This means that the transcript gives no information about x
- How does Victor simulate a transcript?
  - (Witness, challenge, response)

Simulator

- Need to fake a transcript $(g^{w'}, c', r')$
- Simulator:
  - Trick: do not follow the protocol order!
  - First pick the challenge c'
  - Then pick a random response r'
    - Then note that the response must satisfy:
      $g^{r'} = (g^x)^{c'} g^{w'} \rightarrow g^{w'} = g^{r'} / (g^x)^{c'}$
  - Solve for $g^{w'}$
- Proof technique for ZK
  - but also important in constructions (OR)

Non-interactive proof?

- Schnorr's protocol
  - Requires interaction between Peggy and Victor
  - Victor cannot transfer proof to convince Charlie
    - (In fact we saw he can completely fake a transcript)
- **Fiat-Shamir Heuristic**
  - $H[\cdot]$ is a cryptographic hash function
  - Peggy sets $c = H[g^w]$
  - Note that the simulator cannot work any more
    - g^w has to be set first to derive c
- Signature scheme
  - Peggy sets $c = H[g^w, M]$

Generalise to DL represenations

- Traditional Schnorr
  - For fixed g, p and public key $h = g^x$
  - Peggy proves she knows x such that $h = g^x$
- General problem
  - Fix prime p, generators $g_1, \ldots, g_l$
  - Public key $h' = g_1^{x_1} g_2^{x_2} \ldots g_l^{x_l}$
  - Peggy proves she knows $x_1, \ldots, x_l$ such that $h' = g_1^{x_1} g_2^{x_2} \ldots g_l^{x_l}$

DL represenation – protocol

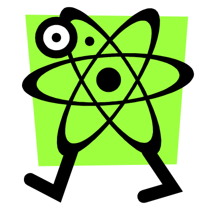
Public: g, p

Knows: $x_1, \ldots, x_l$

Peggy
(Prover)

l random: w_i

Knows:
$h = g_1^{x_1} g_2^{x_2} \ldots g_l^{x_l}$



Victor
(Verifier)

P->V: $\prod_{0<i<l} g^{w_i} = a$ (witness)

V->P: c (challenge)

$r_i = cx_i + w_i$

P->V: $r_1, \ldots, r_l$ (response)

Check:

$$\left(\prod_{0<i<l} g_i^{r_i}\right) = h^c a$$

Let's convince ourselves: $\left(\prod_{0<i<l} g_i^{r_i}\right) = \left(\prod_{0<i<l} g_i^{x_i}\right)^c \left(\prod_{0<i<l} g^{w_i}\right) = h^c a$

DL represenation vs. Schnorr

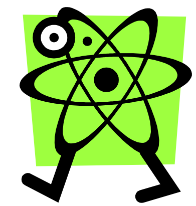
Public: g, p

Knows: $x_1, \ldots, x_l$

Knows: $h = g_1^{x_1} g_2^{x_2} \ldots g_l^{x_l}$

Peggy
(Prover)

l random: w_i



Victor
(Verifier)

P->V: $\prod_{0<i<l} g^{w_i} = a$ (witness)

V->P: c (challenge)

$r_i = cx_i + w_i$

P->V: $r_1, \ldots, r_l$ (response)

Check:

$$\left(\prod_{0<i<l} g_i^{r_i}\right) = h^c a$$

Lets convince ourselves: $\left(\prod_{0<i<l} g_i^{r_i}\right) = \left(\prod_{0<i<l} g_i^{x_i}\right)^c \left(\prod_{0<i<l} g^{w_i}\right) = h^c\, a$

Credentials – showing

- Relation to DL representation
- Credential representation:
  - Attributes x_i
  - Credential $h = g_1^{x_1} g_2^{x_2} \dots g_l^{x_l}$, $\mathrm{Sig}_{\mathrm{Issuer}}(h)$
- Credential showing protocol
  - Peggy gives the credential to Victor
  - Peggy proves a statement on values x_i
    - $X_{age} = 28$ AND $x_{city} = H[\text{Cambridge}]$
  - Merely DL rep. proves she knows x_i

Linear relations of attributes (1)

- Remember:
  - Attributes x_i , $i = 1,\dots,4$
  - Credential $h = g_1^{x_1} g_2^{x_2} g_3^{x_3} g_4^{x_4}$, $Sig_{Issuer}(h)$
- Example relation of attributes:
  - $(x_1 + 2x_2 - 10x_3 = 13)$ AND $(x_2 - 4x_3 = 5)$
  - Implies: $(x_1 = 2x_3 + 3)$ AND $(x_2 = 4x_3 + 5)$
  - Substitute into h
    - $h = g_1^{2x_3+3} g_2^{4x_3+5} g_3^{x_3} g_4^{x_4} = (g_1^3 g_2^5)(g_1^2 g_2^4 g_3)^{x_3} g_4^{x_4}$
    - Implies: $h / (g_1^3 g_2^5) = (g_1^2 g_2^4 g_3)^{x_3} g_4^{x_4}$

Linear relations of attributes (2)

- Example (continued)
  - $(x_1 + 2x_2 - 10x_3 = 13)$ AND $(x_2 - 4x_3 = 5)$
  - Implies: $h / (g_1^3 g_2^5) = (g_1^2 g_2^4 g_3)^{x_3} g_4^{x_4}$
- How do we prove that in ZK?
  - DL representation proof!
    - $h' = h / (g_1^3 g_2^5)$
    - $g_1' = g_1^2 g_2^4 g_3$ $\qquad$ $g_2' = g_4$
  - Prove that you know x_3 and x_4 such that $h' = (g_1')^{x_3} (g_2')^{x_4}$

DL rep. – credential show example

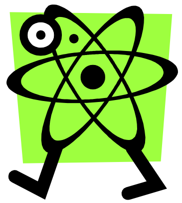
Public: g, p

Knows: x_1, x_2, x_3, x_4

Peggy
(Prover)

random: w_1, w_2

Knows:
$h = g_1^{x_1} g_2^{x_2} g_3^{x_3} g_4^{x_4}$



Victor
(Verifier)

P->V: $g_1'^{w_1} g_2'^{w_2} = a'$ (witness)

V->P: c (challenge)

P->V: r_1, r_2 (response)

$r_1 = cx_3 + w_1$

$r_2 = cx_4 + w_2$

Check:

$$(g_1')^{r_1} (g_2')^{r_2} = (h')^c a$$

Check $(g_1')^{r_1} (g_2')^{r_2} = (h')^c a$

- Reminder
  - $h = g_1^{x_1} g_2^{x_2} g_3^{x_3} g_4^{x_4}$
  - $h' = h / (g_1^3 g_2^5)$ $\quad g_1' = g_1^2 g_2^4 g_3$ $\quad g_2' = g_4$
  - $a = g_1'^{w_1} g_2'^{w_2}$ $\quad r_1 = cx_3 + w_1$ $\quad r_2 = cx_4 + w_1$
- Check:
  - $(g_1')^{r_1} (g_2')^{r_2} = (h')^c a \Rightarrow$
    $(g_1')^{(\cancel{c}x_3+\cancel{w}_1)} (g_2')^{(\cancel{c}x_4+\cancel{w}_1)} = (h / (g_1^3 g_2^5))^{\cancel{c}} \cancel{g_1}'^{w_1} \cancel{g_2}'^{w_2} \Rightarrow$
    $(g_1^{2x_3+3} g_2^{4x_3+5} g_3^{x_3} g_4^{x_4}) = h$

x_1 $\quad x_2$

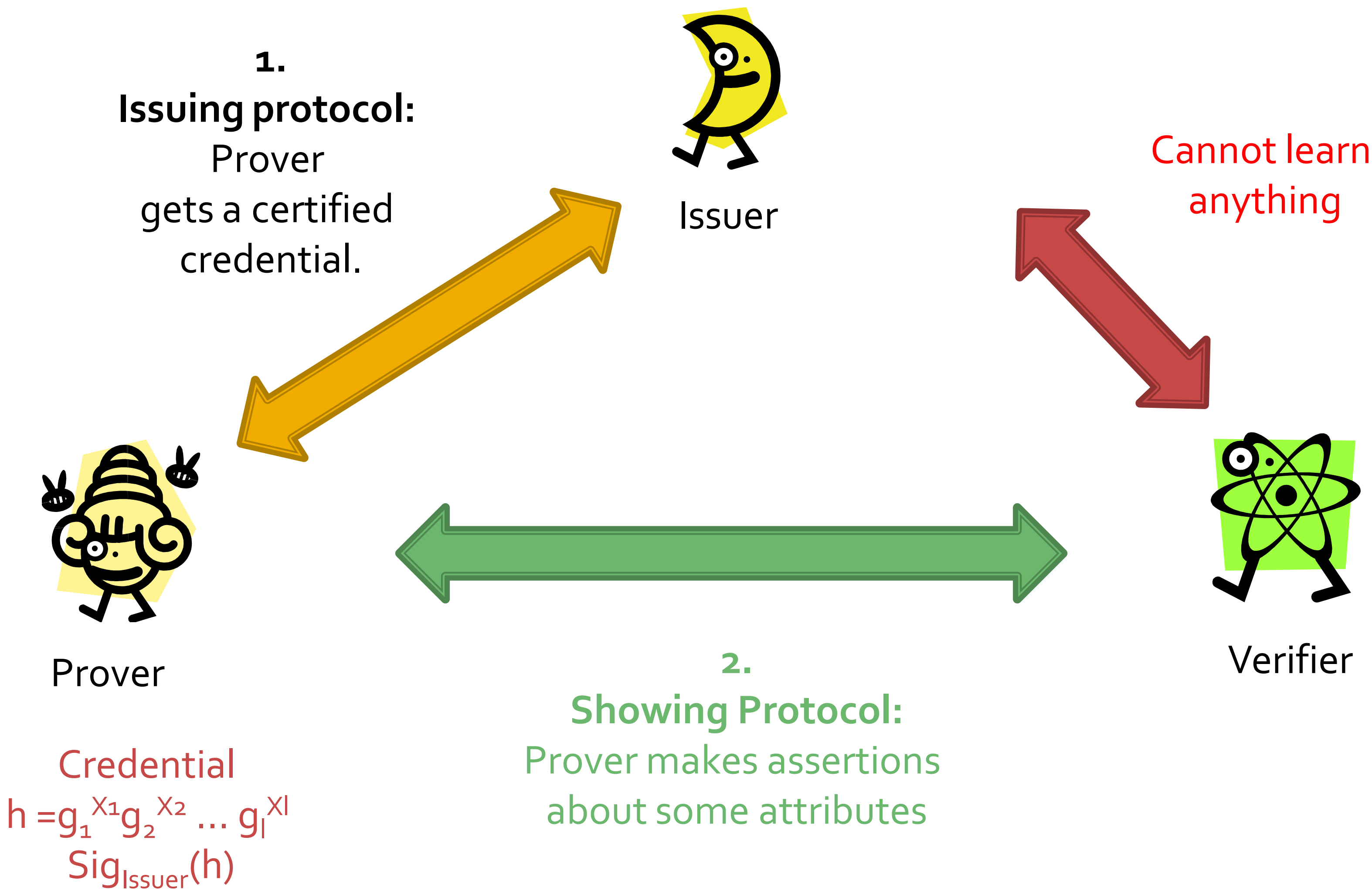
A few notes

- Showing any relation implies knowing all attributes.
- Can make non-interactive (message m)
  - c = H[h, m, a']
- Other proofs:
  - (OR) connector *(simple concept)*
    - (x_{age}=18 AND x_{city}=H[Cambridge]) OR (x_{age}=15)
  - (NOT) connector
  - Inequality (x_{age} > 18) (Yao's millionaire protocol)

Summary of key concepts (1)

- Standard tools
  - Schnorr – ZK proof of knowledge of discrete log.
  - DL rep. – ZK proof of knowledge of representation.
- Credential showing
  - representation + certificate
  - ZK proof of linear relations on attributes (AND)
  - More reading: (OR), (NOT), Inequality

Issuing credentials



Issuing security

- Prover cannot falsify a credential
- Unlinkability
  - Issuer cannot link a showing transcript to an instance of issuing
  - h, $Sig_{issuer}(h)$ have to be unlinkable to issuing
- Achieving unlinkability
  - Issuer's view: $h = g_1^{x_1} g_2^{x_2} \ldots g_l^{x_l}$
  - Prover uses: $h' = g_1^{x_1} g_2^{x_2} \ldots g_l^{x_l} g_0^{a_1}$

Issuing protocol – gory details

Knows: $x_1, x_2, \ldots, x_l$

Public: g, p

Knows: $x_1, x_2, \ldots, x_l$

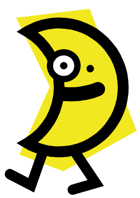
Prover

Private: $x_o, (y_1, \ldots, y_l)$

Public: $h_o = g^{x_o}, g_i = g^{y_i}$

Rand: w_o

I->P: $g^{w_o} = a_o$ (witness)



Issuer

Rand: a_1, a_2, a_3

$h' = h \cdot g^{a_1}$

$c'_o = H[h', g^{a_2}(h_o h)^{a_3} a_o]$

$c_o = c'_o + a_3$

P->I: c_o (challenge)

$r_o = c_o(x_o + \sum_i x_i y_i) + w_o$

I->P: r_o (response)

Check: $g^{r_o} = (h_o h)^{c_o} a_o$

$r'_o = r_o + a_2 + c'_o a_1$

Credential: $h' = g^{a_1} \prod g_i^{x_i}$

Signature: (c'_o, r'_o)

Check: $c'_o = H[h', g^{r'_o}(h_o h')^{-c'_o}]$

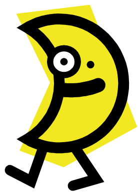
Issuing protocol – Issuer side

Knows: $x_1, x_2, \ldots, x_l$

Public: g, p

Private: $x_o, (y_1, \ldots, y_l)$
Public: $h_o = g^{x_o}, g_i = g^{y_i}$
Rand: w_o

I->P: $g^{w_o} = a_o$ (witness)



~~Non interactive signature: $c_o = H[h, a_o]$~~

Issuer

P->I: c_o (challenge)

$r_o = c_o(x_o + \sum_i x_i y_i) + w_o$

I->P: r_o (response)

ZK knowledge proof of the representation of $h_o h = g^{x_o} \prod g_i^{x_i} = g^{(x_o + \sum_i x_i y_i)}$: Just Schnorr !

Issuing protocol – Prover side (1)

Public: $g, p, h_o = g^{x_o}, g_i = g^{y_i}$

Knows: $x_1, x_2, \dots, x_l$

Prover

I->P: $g^{w_o} = a_o$ (witness)

P->I: c_o (challenge)

I->P: r_o (response)

Rand: a_1, a_2, a_3

$h' = h \cdot g^{a_1}$

$c'_o = H[h', g^{a_2}(h_o h)^{a_3} a_o]$

$c_o = c'_o + a_3$

Check: $g^{r_o} = (h_o h)^{c_o} a_o$

$r'_o = r_o + a_2 + c'_o a_1$

Schnorr Verification:

Issuer knows the representation of $(h_o h)$!

Credential: $h' = g^{a_1} \prod g_i^{x_i}$

Signature: (c'_o, r'_o)

Check: $c'_o = H[h', g^{r'_o}(h_o h')^{-c'_o}]$

Issuing protocol – Prover side (2)

Public: $g, p, h_o = g^{x_o}, g_i = g^{y_i}$

Knows: $x_1, x_2, \ldots, x_l$

Prover

I->P: $g^{w_o} = a_o$ (witness)

Rand: a_1, a_2, a_3

$h' = h \cdot g^{a_1}$

$c'_o = H[h', g^{a_2}(h_o h)^{a_3} a_o]$

$c_o = c'_o + a_3$

1) Set c_o

P->I: c_o (challenge)

I->P: r_o (response)

2) Get r_o such that…

Check: $g^{r_o} = (h_o h)^{c_o} a_o$

$r'_o = r_o + a_2 + c'_o a_1$

Unlinkable

Credential: $h' = g^{a_1} \prod g_i^{x_i}$

Signature: (c'_o, r'_o)

Check: $c'_o = H[h', g^{r'_o}(h_o h')^{-c'_o}]$

My Goal

Issuing protocol – Prover side (3)

Public: $g, p, h_o = g^{x_o}, g_i = g^{y_i}$

Knows: $x_1, x_2, \ldots, x_l$

I->P: $g^{w_o} = a_o$ (witness)

Rand: a_1, a_2, a_3

$h' = h \cdot g^{a_1}$

$\underline{c'_o = H[h', g^{a_2}(h_o h)^{a_3} a_o]}$

$c_o = c'_o + a_3$

P->I: c_o (challenge)

I->P: r_o (response)

Check: $g^{r_o} = (h_o h)^{c_o} a_o$

$r'_o = r_o + a_2 + c'_o a_1$

?

Credential: $h' = g^{a_1} \prod g_i^{x_i}$

Signature: (c'_o, r'_o)

Check: $\underline{c'_o = H[h', g^{r'_o}(h_o h')^{-c'_o}]}$

Check

- Goal:
  - $c'_o = H[h', g^{a_2}(h_o h)^{a_3} a_o] = H[h', g^{r'_o}(h_o h')^{-c'_o}]$
    - So $g^{a_2}(h_o h)^{a_3} a_o = g^{r'_o}(h_o h')^{-c'_o}$ must be true
  - Lets follow:
    - $g^{r'_o}(h_o h')^{-c'_o} = g^{a_2}(h_o h)^{a_3} a_o \Leftrightarrow$

      Substitute r'_o and c'_o
    - $g^{(r_0 + a_2 + c'_0 a_1)} (h_o h)^{-(c_0+a_3)} g^{-c_0 a_1} = g^{a_2}(h_o h)^{a_3} a_o \Leftrightarrow$
    - $(g^{r_0}(h_o h)^{-c_0}) \cancel{(g^{a_2}(h_o h)^{a_3})} = \cancel{(g^{a_2}(h_o h)^{a_3})}\, a_o \Leftrightarrow$

TRUE

Unlinkability

- Issuer sees: c_o, r_o, h
  - Such that $g^{r_o} = (h_o h)^{c_o} a_o$
- Verifier sees: c'_o, r'_o, h'
- Relation:
  - Random: a_1, a_2, a_3
    - $h' = h \cdot g^{\mathbf{a_1}}$
    - $c_o = c'_o + \mathbf{a_3}$
    - $r'_o = r_o + \mathbf{a_2} + c'_o a_1$
- Even if they collude they cannot link the credential issuing and showing

Notes on issuer

- Authentication between Issuer and Peggy
  - Need to check that Peggy has the attributes requested
- Issuing protocol should not be run in parallel!
  - (simple modifications are required)

Full credential protocol

- Putting it all together:
  - Issuer and Peggy run the issuing protocol.
    - Peggy gets:

Credential: $h' = g^{a_1} \prod g_i^{x_i}$

Signature: (c'_0, r'_0)
Check: $c'_0 = H[h', g^{r'_0}(h_0 h')^{-c'_0}]$

  - Peggy and Victor run the showing protocol
    - Victor checks the validity of the credential first
    - Peggy shows some relation on the attributes
      - (Using DL-rep proof on h')

Key concepts so far (2)

- Credential issuing
  - Proof of knowledge of DL-rep & x_o of issuer
  - Peggy assists & blinds proof to avoid linking
- Further topics
  - Transferability of credential
  - Double spending

Key applications

- Attribute based access control
- Federated identity management
- Electronic cash
  - (double spending)
- Privacy friendly e-identity
  - Id-cards & e-passports
- Multi-show credentials!

References

- Core:
  - Claus P. Schnorr. **Efficient signature generation by smart cards**. Journal of Cryptology, 4:161—174, 1991.
  - Stefan Brands. **Rethinking public key infrastructures and digital certificates – building in privacy**. MIT Press.
- More:
  - Jan Camenisch and Markus Stadler. **Proof systems for general statements about discrete logarithms.** Technical report TR 260, Institute for Theoretical Computer Science, ETH, Zurich, March 1997.
  - Jan Camenisch and Anna Lysianskaya. **A signature scheme with efficient proofs.** (CL signatures)

OR proofs

- Peggy wants to prove (A OR B)
  - Say A is true and B is false
- Simple modification of Schnorr
  - Peggy sends witness
  - Victor sends commitment c
  - Peggy uses simulator for producing a response r_B for B
    - (That sets a particular c_B)
    - Peggy chooses c_A such that $c = c_A + c_B$
  - Then she produces the response r_A for A
- Key concept: simulators are useful, not just proof tools!

Strong(er) showing privacy

- Designated verifier proof
  - A OR knowledge of verifier's key
  - Simulate the second part
  - Third parties do nor know if A is true or the statement has been built by the designated verifier!
- Non-interactive proof not transferable!