Authentication without Identification

Jan Camenisch
IBM Research - Zurich
We all increasingly conduct our daily tasks electronically

....are becoming increasingly vulnerable to cybercrimes
33% of cyber crimes, including identity theft, take less time than to make a cup of tea.
10 Years ago, your identity information on the black market was worth $150. Today....
Facts

$4,500,000,000 cost of identity theft worldwide
Houston, we have a problem!
Houston, we have a problem!

“Buzz Aldrin's footprints are still up there”
(Robin Wilton)
Computers don't forget

- Apps built to use & generate (too much) data
- Data is stored by default
- Data mining gets ever better
- New (ways of) businesses using personal data

- Humans forget most things too quickly
- Paper collects dust in drawers
Where's all my data?

The ways of data are hard to understand

- Devices, operating systems, & apps are getting more complex and intertwined
  - Mashups, Ad networks
  - Machines virtual and realtime configured
  - Not visible to users, and experts
  - Data processing changes constantly

→ No control over data and far too easy to loose them
You have no privacy, get over it .....?!?

- Huge security problem!
  - Millions of hacked passwords (100'000 followers $115 - 2013)
  - Stolen credit card number ($5 - 2013)
  - Lots of not reported issues (industrial espionage, etc)

- Difficult to put figures down
  - Credit card fraud
  - Spam & marketing
  - Manipulating stock ratings, etc..

- We know secret services can do it easily, but they are not the only ones
  - this is not about homeland security
  - and there are limits to the degree of protection that one can achieve

- … end we have not event discussed social issues such as democracy etc

- last but not least: data are the new money, so need to be protected
What can we do?

- Most of the **technology** is there (but we have to use it)

- Most of the **legislation** is there (but we have to enforce it)

- **Public awareness** is growing (but we have to feed it)

→ **Privacy by Design!**
of course there are limits...

- tracing is so easy
  - each piece of hardware is quite unique
  - log files everywhere

- .... but that's not the point!
  - it's not about NSA et al.
  - active vs passive “adversaries”

so, still, privacy by design!
The real problem

Applications are designed with the sandy beach in mind but are then built on the moon.

- Feature creep, security comes last, if at all
- Everyone can do apps and sell them
- Networks and systems hard not (well) protected
Security & Privacy is not a lost cause!

We need paradigm shift &
build stuff for the moon
rather than the sandy beach!
That means:

- Reveal only minimal data necessary
- Encrypt every bit
- Attach usage policies to each bit

Cryptography can do that!
Business Case for Privacy

- Business case always depends on specific example
- The bad news: data is money – so every one wants it
  - Big data, sensors, etc.
  - Marketing
- No data is easier to protect than lots of data
  - e.g., losing credit card number
- Privacy is part of security
  - identity theft
  - getting access to and manipulating systems, etc.
- Laws are an important driver
  - like many other technical areas, cf. cars
Privacy by design....

@ Communication layer
- TOR, JAP, etc

@ Authentication layer
- privacy-preserving attribute-based credentials

@ Application layer
- eVoting, ePolls, ....
- all apps should be done as “privacy by design”
Anonymous communication

e.g., mix networks, onion routing (TOR), DC-nets

Bandwidth suffers

Physical layer notoriously hard to protect
Privacy at the Authentication Layer
Privacy Friendly Identity Management
What is an identity?

- name
- salary
- credit card number
- hobbies
- phone number
- address
- language skills
- marital status
- birth date
- insurance
- blood group
- health status
- leisure
- work
- public authority
- shopping
- health care
- nick name
Identities – Identity Management

- **ID**: set of attributes shared w/ someone
  - attributes are not static: user & party can add

- **ID Management**: two things to make ID useful
  - authentication means
  - means to transport attributes between parties

- **Control attributes with policies**:
  - define requested data
  - define allowed usage (audience)

- **Privacy by design**
  - minimal amount of data

→ Privacy-preserving attribute-based credentials
Identities
Authentication and attribute transfer
Alice wants to watch a movie at Movie Streaming Service

I wish to see Alice in Wonderland
Alice wants to watch a movie at Movie Streaming Service

You need:
- subscription
- be older than 12
Watching the movie with the traditional solution

Using digital equivalent of paper world, e.g., with X.509 Certificates

ok, here's
- my eID
- my subscription

Alice

Movie Streaming Service
Watching the movie with the traditional solution

...with X.509 Certificates

Aha, you are
- Alice Doe
- born on Dec 12, 1975
- 7 Waterdrive
- CH 8003 Zurich
- Married
- Expires Aug 4, 2018

Mplex Customer
- #1029347
- Premium Subscription
- Expires Jan 13, 2016

Movie Streaming Service
Watching the movie with the traditional solution

This is a privacy and security problem!
- identity theft
- discrimination
- profiling, possibly in connection with other services

Aha, you are
- Alice Doe
- born on Dec 12, 1975
- 7 Waterdrive
- CH 8003 Zurich
- Married
- Expires Aug 4, 2018

Mplex Customer
- #1029347
- Premium Subscription
- Expires Jan 13, 2016

Movie Streaming Service
Authentication – Traditional approach

ID Management: make ID useful
- ID comes w/ authentication means

- Transport of attributes

- Alice has public and private key
- Can use cryptographic identification protocols to authenticate as owner

- Certificate issued by one party, certificate contains attributes
- Certificate is verified by other party w.r.t. Issuing party's public key (public verification key)

Is really just paper world made electronic!
credential / certificate
- signed list of attribute-value pairs

Name = Alice Doe
Birth date = 1997/01/26
Public key = 98a9d8ac9237..

signed by the issuer
With traditional PKI, e.g., X.509 v3 certificates

Privacy: - have to disclose all attributes in certificate
  - public key as unique identifier

Security: - verifier's collection of attributes
  - target for identity thieves

name = Alice Doe,
birth date = 1973/01/26,
name = Alice Doe,
birth date = 1973/01/26,
Using a certificate again…

With traditional PKI, e.g., X.509 v3 certificates
Watching the movie with the traditional solution

With OpenID (similar protocols), e.g., log-in with Facebook

ok, I'm Alice@facebook.com

Alice

Movie Streaming Service
Watching the movie with the traditional solution

With OpenID and similar solution, e.g., log-in with Facebook

Aha, Alice is watching a 12+ movie
Watching the movie with the traditional solution

With OpenID and similar solution, e.g., log-in with Facebook

Aha, Alice is watching a 12+ movie

Aha, you are
- Alice@facebook.com
- 12+
  Mplex Customer
- #1029347
- Premium Subscription
- Expires Jan 13, 2016

Movie Streaming Service
Authentication – Online Solution

e.g., SAML, WS-Federation, OpenID, Facebook Connect

Privacy: issuer knows who visits which website at which time (often from own records, if not by correlating logs with website)

Security: issuance key online 24/7
best of both worlds

- Privacy:
  - unlinkable transactions
  - minimal information disclosure

- Security:
  - offline issuer

Examples include IBM Identity Mixer and Microsoft U-Prove
Privacy-protecting authentication with Privacy ABCs

Users' Keys:
- One secret Identity (secret key)
- Many Public Pseudonyms (public keys)

Concept:
- Key binding
- Pseudonyms (and domain pseudonyms)

→ use a different identity for each communication partner or even transaction
Two kinds of pseudonyms

- Regular
- Scope exclusive (also called domain pseudonym)

Specification of pseudonym very generic, still:

- Scope: String
- Exclusive: Boolean
- PseudonymValue: AnyValue

For regular pseudonym, scope is non-binding description

```xml
<abc:Pseudonym Scope="xs:string"?
Exclusive="xs:boolean"?>

  <abc:PseudonymValue>...</abc:PseudonymValue>
</abc:Pseudonym>
```
Certified attributes from Identity provider

- Issuing a credential

Name = Alice Doe
Birth date = April 3, 1997

Concept:
- Credentials
Credential Specification

Specification ID: URN

NumberOfAttributes: INT

List of Attributes, each consisting of:

- Type: URN
  - Value: first name
- DataType: URN
  - Value: string
- Encoding: URN
  - Value: sha256

KeyBinding: true

Revocation: false

Credentials is essentially the same extended with:

Each attribute consists additionally

- Value: DataType

Crypto Value: AnyValue (according to alg.; digital signature)
Example of Credential Specification

```xml
<CredentialSpecification Version="1.0" KeyBinding="true" Revocable="true">


  <AttributeDescriptions MaxLength="32">
    <AttributeDescription Type="http://abc4trust.eu/wp6/credspec/credSchool/firstName"
      DataType="xs:string" Encoding="abc:sha256"/>
    <AttributeDescription Type="http://abc4trust.eu/wp6/credspec/credSchool/lastName"
      DataType="xs:string" Encoding="abc:sha256"/>
    <AttributeDescription Type="http://abc4trust.eu/wp6/credspec/credSchool/civicNr"
      DataType="xs:integer" Encoding="abc:plain"/>
    <AttributeDescription Type="http://abc4trust.eu/wp6/credspec/credSchool/gender"
      DataType="xs:boolean" Encoding="abc:zero-one"/>
    <AttributeDescription Type="http://abc4trust.eu/wp6/credspec/credSchool/school"
      DataType="xs:string" Encoding="xenc:sha256"/>
  </AttributeDescriptions>

</CredentialSpecification>
```
Certified attributes from purchasing department

- Issuing a credential
Privacy-protecting authentication with Privacy ABCs

I wish to see Alice in Wonderland

You need:
- subscription
- be older than 12

Concept:
- Presentation policy
Presentation policy: which attributes certified by whom a verifier requires to grant access

- Credential Alias = String
  - Possible Credentials: {Credential Spec}
  - Possible Issuers: {IssuerParameters}
  - {Disclosed Attributes: AttributeType}
    - Possible Inspectors: {InspectorPublicKey}
    - Inspection Grounds
  - SameKeyBindingAs: String

- AttributePredicate
  - Function: definedFunctions
  - Attribute: CredentialAlias, AttributeType
  - ConstantValue: AnyValue

- Message: String
  - DateGreaterThan: 1987-03-05
“reveal civic number from school credential”

```xml
<PresentationPolicyAlternatives>
  <PresentationPolicy PolicyUID="revealCivicNr">
    <Message>
      <Nonce>bkQydHEQWDR4TUZzbXJKYUphdVM=</Nonce>
    </Message>
    <Credential Alias="schoolcred">
      <CredentialSpecAlternatives>
        <CredentialSpecUID>http://abc4trust.eu/wp6/credspec/credschool</CredentialSpecUID>
      </CredentialSpecAlternatives>
    </Credential>
  </PresentationPolicy>
</PresentationPolicyAlternatives>
```

User  presentation policy  presentation token  Verifier
Proving identity claims
- but does *not* send credentials
- only minimal disclosure

*Concept:*
- Presentation token

- valid subscription
- eID with age ≥ 12
Proving identity claims
- but does not send credential
- only minimal disclosure

Transaction is not linkable to any other of Alice's transactions!

Aha, you are
- older than 12
- have a subscription

(Public Verification Key of issuer)
Proving Identity Claims: Minimal Disclosure

**Alice Doe**
- Dec 12, 1998
- Hauptstr. 7, Zurich, CH
- Single
- Exp. Aug 4, 2018

**Verified ID**

**Age: 12+**
- Exp. Valid

**Verified ID**
• All (key bound) credentials are bound to same secret key
• For presentation, different credentials can be combined, correct combination is assured by key binding
All transaction are unlinkable by default!

Presentation tokens and pseudonyms are unlinkable!
Recall Concepts and Terms

- Pseudonym
- Credential
- Presentation token
- Presentation policy

(Issuer parameter)

Credential specification

12 < age ?

(Verifier parameter)
Summary of Properties

- Protection of user's privacy
  - unlinkability (multi-use)
  - using/combining multiple credentials
  - selective disclosure
  - predicated over attributes

- Strong authentication
  - unforgeability of presentation tokens
    - Wrong statements
    - Possession of credentials not owned
    - Combination of credentials from different users
Private credentials vs. classical ones

- each user has a single secret key but many public keys
- attributes might be hidden from issuer

Certificate = signature on user's secret key + attributes

- prove knowledge of certificate
  - prove that different certificates contain same secret key
  - selectively reveal attributes

Certificate = signature on user's public key + attributes

- reveal certificate
Try Identity Mixer for yourself

Try yourself:  
Build your app:  
Source code:  
Info:  
→ idemixdemo.mybluemix.net  
→ github.com/IBM-Bluemix/idemix-issuer-verifier  
→ github.com/github.com/p2abcengine/p2abcengine  
→ ibm.biz/identity_mixer
User – Verifier: architectural view [abc4trust.eu]

Available on github.com/p2abcengine & abc4trust.eu/idemix
IBM's Privacy ABCs: Identity Mixer

- Scientific foundation laid 15 years ago, well studied & award winning
- Successful real-world pilots in series of EU projects

You can have identity mixer, too!
- Open-source implementation: https://github.com/p2abcengine
- Idemix-as-a-Service on IBM Bluemix
- Web-based demo to try for everyone
- Coming soon: Idemix on mobile
There is more
Here's just a few examples
If car is broken: ID with insurance needs be retrieved
Can verifiably encrypt any certified attribute \((optional)\)
TTP is off-line & can be distributed to lessen trust
If Alice was speeding, license needs to be revoked!

There are many different use cases and many solutions

- Variants of CRL work (using crypto to maintain anonymity)
  - Accumulators
  - Signing entries & Proof, ....
- Limited validity – certs need to be updated
- ... For proving age, a revoked driver's license still works
Limits of anonymity possible (*optional*):

If Alice and Eve are on-line together they are caught!

Use Limitation – anonymous until:

* If Alice used certs > 100 times total...
* ... or > 10,000 times with Bob

Alice's cert can be bound to hardware token (e.g., TPM)
Proving 12+, 18+, 21+ without disclosing the exact date of birth – privacy and compliance with age-related legislation

- Movie streaming services
- Gaming industry
- Online gambling platforms
- Dating websites
- Social benefits for young/old people
Use Case: Anonymous Authentication

Use certificate anonymously
Use Case: Anonymous Authentication

Use certificate anonymously

Server shall be unable to tell whether or not it is the same user
Use Case: Anonymous Access to a Database

Simple case: DB learns not who accesses DB
Better: Oblivious Access to Database (OT with AC)

• Server must not learn *who* accesses
• *which* record
• Still, Alice can access only records she is *authorized* for
Who accesses *which data* at which time can reveal sensitive information about the users (their research strategy, location, habits, etc.)

- Patent databases
- DNA databases
- News/Journals/Magazines
- Transportation: tickets, toll roads
- Loyalty programs
Anonymous consultations with specialists

- online chat with at physician / online consultation with IBM Watson to check eligibility

1. Alice proves she has insurance
2. Alice describes symptoms
3. Alice gets credential that she is allowed to get treatment
4. Alice gets treatment from physician, hospital, etc
5. Alice sends bill to insurance and proves that she had gotten the necessary permission for the treatment.
**Payment Use Case**

- Credential = Bank note
- Double spending need to be prevented/detected
  - On-line or Off-line modi possible
- Money laundering can also be taken care of
The credit card data is never revealed to the merchant, only to the credit card provider

- Bank issues a classic credit card
- User registers at a special portal to obtain the Identity Mixer credential
- User derives a token allowing that store to withdraw the money
- Users cannot be linked across purchases/shops
- Stored credit card info useless to hackers!
Polls, recommendation platforms

Providing anonymous, but at the same time legitimate feedback

- **Online polls**
  - applying different restrictions on the poll participants: location, citizenship

- **Rating and feedback platforms**
  - anonymous feedback for a course only from the students who attended it
  - wikis
  - recommendation platforms
Scenario:

- Pollster(s) and a number of users
- Only registered user (e.g., students who took a course) can voice opinion (e.g., course evaluation)
- User can voice opinion only once (subsequent attempts are dropped)
- Users want to be anonymous
- In different polls user must not linkable
- User generates pseudonym (ID for registration)
- User obtains credential on pseudonym stating that she is eligible for polls
- Credential can contain attributes (e.g., course ID) about her
Use Case Polling – Solution: Submit Poll

1. User generates *domain* pseudonym, domain = pollID

2. User transforms credential

3. Transformed credential with a subset of the attributes
   - User is anonymous and unlinkable
   - Multiple opinions are detected because uniqueness of domain pseudonym
Implementation of concepts
Efficient realization requires special algorithms

- *Commitment scheme* for pseudonyms
  - One secret key, many public keys

- *Signature scheme* for credentials
  - Normal signature schemes would be inefficient
  - Needs to allow for
    - Signing attributes separately
    - Efficient transformation into presentation token

- *Zero-knowledge proofs* of knowledge for inspection
  - Needs to be compatible with signature and commitment scheme

- *Verifiable encryption* scheme for inspection
  - Needs to be compatible with other schemes

- *Certificate revocation* lists won't work → special schemes
A set \( G \) with operation \( \circ \) is called a **group** if:

- **Closure**
  
  for all \( a, b, \) in \( G \) \( \rightarrow a \circ b \) in \( G \)

- **Associativity**
  
  for all \( a, b, c, \) in \( G \) \( \rightarrow (a \circ b) \circ c = a \circ (b \circ c) \)

- **Identity**
  
  there exist some \( e \) in \( G \), s.t. for all \( a \): \( a \circ e = a \)

- **Invertibility**
  
  for all \( a \) in \( G \), there exist \( a^{-1} \) in \( G \): \( a \circ a^{-1} = e \)

Most used in cryptography: multiplication of integers mod \( p \) (with \( p \) prime)
**Excurus: Number Theory | Discrete Logarithm**

exponentiation = repeated application of $\cdot$, e.g., $a^3 = a \cdot a \cdot a$

given $g, x$ it is easy to compute $g^x, g^{1/x}, \ldots$

given $g^x, g^y$ it is easy to compute $g^x g^y = g^{x+y}$

Discrete Log Assumption
- given $g^x$ it is hard to compute $x$

Diffie-Hellman Assumption
- given $g^x$ and $g^y$ it is hard to compute $g^{xy}$

Decisional Diffie-Hellman Assumption
- given $g^x, g^y, \text{ and } g^z$ it is hard to decide if $g^z = g^{xy}$
zero-knowledge proofs
Zero-Knowledge Proofs

- interactive proof between a prover and a verifier about the prover's knowledge

properties:

zero-knowledge
verifier learns nothing about the prover's secret

proof of knowledge (soundness)
prover can convince verifier only if she knows the secret

completeness
if prover knows the secret she can always convince the verifier
Given group $\langle g \rangle$ and element $y \in \langle g \rangle$.

Prover wants to convince verifier that she knows $x$ s.t. $y = g^x$ such that verifier only learns $y$ and $g$.

Prover:

- Random $r$
- $t := g^r$
- $s := r - cx$

Verifier:

- Random $c$
- $t = g^s y^c$?

notation: $\text{PK}\{(\alpha): y = g^\alpha\}$
Signature $SPK((\alpha): \ y = g^\alpha}(m)$:

Signing a message $m$:
- chose random $r \in \mathbb{Z}_q$ and
- compute $c := H(g^r || m) = H(t || m)
  \ s := r - cx \mod (q)$
- output $(c,s)$

Verifying a signature $(c,s)$ on a message $m$:
- check $c = H(g^s y^c || m)$ ?  $\leftrightarrow$  $t = g^s y^c$ ?

Security:
- underlying protocol is zero-knowledge proof of knowledge
- hash function $H(.)$ behaves as a “random oracle.”
Zero Knowledge Proofs of Knowledge of Discrete Logarithms

Non-interactive (Fiat-Shamir heuristic, Schnorr Signatures):

\[ PK\{(\alpha): \ y = g^\alpha \}(m) \]

Many Exponents:

\[ PK\{(\alpha, \beta, \gamma, \delta): \ y = g^\alpha h^\beta z^\gamma k^\delta u^\beta \} \]

Logical combinations:

\[ PK\{(\alpha, \beta): \ y = g^\alpha \land z = g^\beta \land u = g^\beta h^\alpha \} \]
\[ PK\{(\alpha, \beta): \ y = g^\alpha \lor z = g^\beta \} \]

Intervals and groups of different order (under SRSA):

\[ PK\{(\alpha): \ y = g^\alpha \land \alpha \in [A,B] \} \]
\[ PK\{(\alpha): \ y = g^\alpha \land z = g^\alpha \land \alpha \in [0, \min\{\text{ord}(g), \text{ord}(g)\}] \} \]
commitment scheme
Commitment Scheme: Functionality

1. User A sends a commitment to a message \( m \) to a secure storage device.
2. User B requests the commitment \( m \).
3. User A reveals the message \( m \) and the timestamp 2-36-17.
Commitment Scheme: Security

Binding

\[ m, 2-36-17 \]

\[ m', 3-21-11 \]
Commitment Scheme: Security

Binding

\( m, 2-36-17 \)

\( m', 3-21-11 \)

\( m, \epsilon \)

\( m, \epsilon \)

\( m' \)

\( m' \)
Hiding: for all message $m, m'$
Commitment Scheme: Security

Hiding: for all message \( m, m' \)

\( m \rightarrow m \rightarrow m' \rightarrow m' \)

\( m \rightarrow m' \rightarrow ? \rightarrow m' \)
Commitment Schemes

Group $G = \langle g \rangle = \langle h \rangle$ of order $q$

To commit to element $x \in \mathbb{Z}_q$:

- **Pedersen:** perfectly hiding, computationally binding
  
  choose $r \in \mathbb{Z}_q$ and compute $c = g^x h^r$

- **ElGamal:** computationally hiding, perfectly binding:
  
  choose $r \in \mathbb{Z}_q$ and compute $c = (g^x h^r, g^r)$

To open commitment:

- reveal $x$ and $r$ to verifier
- verifier checks if $c = g^x h^r$
Proof of Knowledge of Contents

Proof of Relations among Contents

\[ m = 2 \cdot m' \]

true
Commitment Scheme: Extended Features

Let \( C_1 = g^m h^r \) and \( C' = g^{m'} h^r \) then:

\[
PK\{(\alpha, \beta) : C = g^\beta h^\alpha \}
\]

\[
PK\{(\alpha, \beta, \gamma) : C' = g^\beta h^\alpha \land C = (g^2)^\beta h^\gamma \}
\]
signature schemes
Signature Scheme: Functionality

Key Generation
Signature Scheme: Functionality

\[ \sigma = \text{sig}((m_1, \ldots, m_k)) \]
Signature Scheme: Functionality

 Verification

\[ \sigma = \text{sig}((m_1, \ldots, m_k)) \]

\[ \text{ver}(\sigma, (m_1, \ldots, m_k)) = \text{true} \]
Unforgeability under Adaptive Chosen Message Attack
Unforgeability under Adaptive Chosen Message Attack
Unforgeability under Adaptive Chosen Message Attack

\[ \sigma' \text{ and } m' \neq m_i \text{ s.t. } \]
\[ \text{ver}(\sigma', m', \varnothing) = \text{true} \]
Unforgeability under Adaptive Chosen Message Attack

\[ \sigma' \text{ and } m' \neq m_i \text{ s.t. } \text{ver}(\sigma', m', \mathcal{A}) = \text{true} \]
signature schemes with protocols
Signature Scheme: Signing Hidden Messages

\[
\sigma = \text{sig}((m_1, \ldots, m_j, m_{j+1}, \ldots, m_k))
\]

\[
\text{ver}(\sigma, (m_1, \ldots, m_k),\hat{\sigma}) = \text{true}
\]

Verification remains unchanged!

Security requirements basically the same, but

- Signer should not learn any information about \(m_1, \ldots, m_j\)
- Forgery w.r.t. message clear parts and opening of commitments

\[
(0, \ldots, m_j, m_{j+1}, \ldots, m_k)
\]
Proving Possession of a Signature

\[ \sigma \text{ on } (m_1, \ldots, m_k) \]
Proving Possession of a Signature

σ on (m₁,..., mₖ)
Proving Possession of a Signature

\[ \sigma \text{ on } (m_1, \ldots, m_k) \]

Variation:
- Send also \( m_i \) to verifier and
- Prove that committed messages are signed
- Prove properties about hidden/committed \( m_i \)
Blind Signatures vs Signatures with Protocols

*can be used multiple times*

Damgaard, Camenisch & Lysyanskaya

Strong RSA, DL-ECC, ..

*can be used only once*

Chaum, Brands, et al.

Discrete Logs, RSA, ..
Some signature schemes
RSA Signature Scheme – For Reference

Rivest, Shamir, and Adlemann 1978

Secret Key: two random primes $p$ and $q$
Public Key: $n := pq$, prime $e$, and collision-free hash function $H: \{0,1\}^* \rightarrow \{0,1\}^\ell$

Computing signature on a message $m \in \{0,1\}^*$

$d := 1/e \mod (p-1)(q-1)$

$s := H(m)^d \mod n$

Verification of signature $s$ on a message $m \in \{0,1\}^*$

$s^e = H(m) \pmod{n}$

Correctness: $s^e = (H(m)^d)^e = H(m)^{d\cdot e} = H(m) \pmod{n}$
Verification signature on a message $m \in \{0,1\}^*$

$$s^e := H(m) \pmod{n}$$

Wanna do proof of knowledge of signature on a message, e.g.,

$$\text{PK}\{ (m,s): s^e = H(m) \pmod{n} \}$$

But this is not a valid proof expression!!!! :-(
Public key of signer: RSA modulus $n$ and $a_i, b, d \in \mathbb{QR}_n$.

Secret key: factors of $n$

To sign $k$ messages $m_1, \ldots, m_k \in \{0,1\}^\ell$:

- choose random prime $2^{\ell+2} > e > 2^{\ell+1}$ and integer $s \approx n$
- compute $c$:
  \[
  c = \left( d / (a_1^{m_1} \cdots a_k^{m_k} b^s) \right)^{1/e} \mod n
  \]
- signature is $(c,e,s)$
To verify a signature \((c,e,s)\) on messages \(m_1, ..., m_k\):

- \(m_1, ..., m_k \in \{0,1\}^\ell\):
- \(e > 2^{\ell+1}\)
- \(d = c^e a_1^{m_1} \cdot ... \cdot a_k^{m_k} b^s \mod n\)

Theorem: *Signature scheme is secure against adaptively chosen message attacks under Strong RSA assumption.*
Sign blindly with CL signatures

\[\sigma = \text{sig}(\langle m_1, \ldots, m_j m_{j+1}, \ldots, m_k \rangle, \hat{\sigma})\]

Choose \( e, s'' \)

\[c = (d / (C a_3^{m_3} b^{s''}))^{1/e} \mod n\]

\[d = c^e a_1^{m_1} a_2^{m_2} a_3^{m_3} b^{s''+s''} \mod n\]

\[C = a_1^{m_1} a_2^{m_2} b^{s'}\]
Recall: \[ d = c^e a_1^{m_1} a_2^{m_2} b^s \mod n \]

Observe:
- Let \( c' = c b^\dagger \mod n \) with randomly chosen \( \dagger \)
- Then \( d = c'^e a_1^{m_1} a_2^{m_2} b^{s-e\dagger} \mod n \), i.e., \( (c', e, s^* = s-e\dagger) \) is also signature on \( m_1 \) and \( m_2 \)

To prove knowledge of signature \((c', e, s^*)\) on \( m_2 \) and some \( m_1 \)
- provide \( c' \)
- \( \text{PK}\{(\varepsilon, \mu_1, \sigma) : \ d/a_2^{m_2} := c'^\varepsilon a_1^{\mu_1} b^\sigma \land \mu \in \{0,1\}^\ell \land \varepsilon > 2^{\ell+1} \} \)

\[ \rightarrow \text{proves } d := c'^\varepsilon a_1^{\mu_1} a_2^{m_2} b^\sigma \]
Solving a Use-Case: Polling
Polling: Scenario and Requirements

Scenario:

- Pollster(s) and a number of users
- Only registered user (e.g., students who took a course) can voice opinion (e.g., course evaluation)
- User can voice opinion only once (subsequent attempts are dropped)
- Users want to be anonymous
- A user's opinion in different polls not linkable
Polling – Solution: Registration

- User generates pseudonym (ID for registration)
- User obtains credential on pseudonym stating that she is eligible for polls, i.e., $(c,e,s)$

$$d = c^e a_1^{sk} a_2^{name} b^s \pmod{n}$$

- Credential can contain attributes (e.g., course ID) about her
Polling – Solution: Submit Poll

1. User generates domain pseudonym, domain = pollID

2. User transforms credential

3. Transformed credential with a subset of the attributes
   - User is anonymous and unlinkable
   - Multiple casts are detected because uniqueness of domain pseudonym
1. Domain pseudonym: \( P = g^d^{sk} = H(\text{pollID})^{sk} \)

Ensures privacy: \( P1 = H(\text{pollID1})^{sk} \) and \( P2 = H(\text{pollID2})^{sk} \) are unlinakble

(under the Decisional Diffie-Hellman assumption)

2. User transforms credential:

- \( c' = c b^{s'} \mod n \) with randomly chosen \( s' \)

- \( \text{SPK}((\epsilon, \mu_1, \mu_2, \sigma) : P = g_1^\mu_1 \land d := c'^\epsilon a_1^{\mu_1} a_2^{\mu_2} b^\sigma (\mod n) \land \mu_1, \mu_2 \in \{0,1\}^\ell \land \epsilon > 2^{\ell+1}) \text{ (opinion)} \)
Realizing On-Line eCash
eCash scenario & requirements

Requirements
- Anonymity: Withdrawal and Deposit must be unlinkable
- No Double Spending: Coin is bit-strings, can be spend twice
Towards a Solution: do it like paper money

- Sign notes with digital signature scheme
  - Note = (serial number #, value)
  - Secure because
    - signature scheme can not be forged
    - bank will accepts some serial number only once → on-line e-cash
  - Not anonymous because (cf. paper solution)
    - bit-string of signature is unique
    - serial number is unique
Towards a Solution

- Use (more) cryptography
  - Hide serial number from bank when issuing
    - e.g., sign commitment of serial number
  - Reveal serial number and proof
    - knowledge of signature on
    - commitment to serial number
  - Anonymous because of commitments scheme and zero-knowledge proof
E-Cash
Recall basic idea

- Issue coin: Hide serial number from bank when issuing
  - sign commitment of random serial number

- Spend coin: reveal serial number and proof
  - knowledge of signature on
  - commitment to serial number
On-line E-cash: Withdrawal

Choose $e,s''$

$c = \left(\frac{d}{C b^{s''}}\right)^{1/e} \mod n$

choose random $\#, s'$

and compute

$C = a_1^\# b^{s'}$

$(c,e,s'')$ s.t.

$d = c^e a_1^\# b^{s'' + s'} \pmod{n}$
On-line E-cash: Payment

\[(c, e, s'' + s') \text{ s.t. } d = c^e a_1^# b^{s''} + s' \pmod{n}\]

compute
\[c' = c b^{s'} \pmod{n}\]

proof = \[PK\{(\varepsilon, \mu, \rho, \sigma) : d / a_1^# = c' \varepsilon b^\sigma \pmod{n}\}\]
On-line E-cash: Payment

\[(c, e, s'' + s') \text{ s.t.} \]
\[d = c^e a_1^{#} b^{s'' + s'} \pmod{n} \]

compute
\[c' = c b^{s'} \pmod{n} \]

proof = \[\text{PK}\{(\varepsilon, \mu, \rho, \sigma) : \ d / a_1^{#} = c'^{\varepsilon} b^{\sigma} \pmod{n} \} \]
**Anonymity**
- Bank does not learn \( # \) during withdrawal
- Bank & Shop do not learn \( c, e \) when spending
Double Spending:

- **Spending = Compute**
  
  \[ -c' = c \cdot b^{s'} \mod n \]

- **proof = PK\{ (\varepsilon, \mu, \rho, \sigma) : \frac{d}{a_1^#} = c' \cdot \varepsilon \cdot b^\sigma \mod n \} \]

- **Can use the same # only once....**
  
  - If more #'s are presented than withdrawals:
    
    - Proofs would not sound
    
    - Signature scheme would not secure
Realizing Off-Line eCash
Recall On-Line E-Cash

On-Line Solution:
1. Coin = random serial # (chosen by user) signed by Bank
2. Banks signs blindly
3. Spending by sending # and prove knowledge of signature to Merchant
4. Merchant checks validy w/ Bank
5. Bank accepts each serial # only once.

Off-Line:
- Can check serial # only after the fact
- … but at that point user will have been disappeared...
Towards off-line signatures

Goal:
- spending coin once: OK
- spending coin twice: anonymity is revoked

Seems like a paradox, but crypto is all about solving paradoxical problems :-)
Main Idea:
- Include $\#$, id, $r$ in credential ($\#$, $r$ hidden from bank/issuer)
- Upon spending:
  reveal $\#$ and $t = id + r \cdot u$
  with $u$ randomly chosen by merchant
- $t$ won't reveal anything about id!
- However, given two (or more) equations (for the same $\#$, id, $r$)
  $t_1 = id + r \cdot u_1$
  $t_2 = id + r \cdot u_2$
  one can solve for id.
choose random $\#, r, s'$
and compute

$$C = a_1^\# a_2^r b^{s'}$$

$$d = c^e C a_3^{nym} b^{s''} \mod n$$

$$(c,e,s''+s') \text{ s.t.}$$

$$d = c^e a_1^\# a_2^r a_3^{nym} b^{s''} + s' \pmod{n}$$
Let $G=\langle g \rangle$ be a group of order $q$

$$(c,e,s''+s') \text{ s.t. }$$

$$d = c^e a_1^\# r a_2 a_3^n ym b^s'' + s' \pmod{n}$$

compute

$t = r + u \text{ nym mod q}$

$c' = c b^{s'} \mod n$

proof = $PK\{(\varepsilon, \mu, \rho, \sigma) :$

$$d / a_1^\# = c'\varepsilon a_2^\rho a_3^\mu b^\sigma \pmod{n} \land g^t = g^\rho (g^u)^\mu \}$$
PK\{ (\varepsilon, \mu, \rho, \sigma) : \\
\frac{d}{a_1} \# = c' \varepsilon a_2 \rho a_3 \mu b \sigma \pmod{n} \land g^\dagger = g^\rho (g^u \mu) \}

1. \quad d = c' \varepsilon a_1 \# a_2 \rho a_3 \mu b \sigma \pmod{n} \\
\Rightarrow (c', \varepsilon, \sigma) \text{ is a signature on } \# , \mu, \rho

2. \quad g^\dagger = g^{\rho + u \mu} \\
\Rightarrow \dagger = \rho + u \mu \pmod{q}, \\
\text{i.e., } \dagger \text{ was computed correctly!}
If so:
1. $t = \rho + u \mu \pmod{q}$
2. $t' = \rho + u' \mu \pmod{q}$

solve for $\rho$ and $\mu$.

$\Rightarrow \mu = nym$ because of proof
Off-line E-cash: Security

- Unforgeable:
  - no more coins than #'s,
    - otherwise one can forge signatures
    - or proofs are not sound
  - if coins with same # appears with different u's => reveals nym

- Anonymity:
  - # and r are hidden from signer upon withdrawal
  - t does not reveal anything about nym (is blinded by r)
  - proof proof does not reveal anything
Extensions and more

e-Cash

- K-spendable cash
  - Multiple serial numbers and randomizers per coin
  - Use PRF to generate serial number and randomizers from seed in coin

- Money laundering preventions
  - Must not spend more that $10000 dollars with same party
  - Essentially use additional coin defined per merchant that controls this

Other protocols from these building blocks, essentially anything with authentication and privacy

- Anonymous credentials, eVoting, ...

Alternative building blocks

- A number of signatures scheme that fit the same bill
- (Verifiable) encryption schemes that work along as well
- Alternative framework: Groth-Sahai proofs plus “structure-preserving” schemes
Conclusion
Conclusions

- **Roadmap**
  - Explain possibilities to engineers, policy makers etc
  - Usable prototypes
  - Public infrastructure for privacy protection
  - Laws with teeth (encourage investment in privacy)

- **Challenges**
  - Internet services get paid with personal data (inverse incentive)
  - End users are not able to handle their data (user interfaces..)
  - Security technology typically invisible and hard to sell
  - Solutions seems easier without privacy and security

- **Towards a secure information society**
  - Society changes quickly and gets shaped by technology
  - Consequences are hard to grasp (time will show...)
  - We must inform and engage in a dialog
Thank you!

- eMail: jca@zurich.ibm.com
- Links:
  - www.abc4trust.eu
  - www.PrimeLife.eu
  - www.zurich.ibm.com/idemix
  - github.com/p2abcengine & abc4trust.eu/idemix
References


- Mihir Bellare: Computational Number Theory
  http://www-cse.ucsd.edu/~mihir/cse207/w-cnt.pdf


- Jan Camenisch, Maria Dubovitskaya, Gregory Neven: Oblivious transfer with access control. ACM Conference on Computer and Communications Security 2009: 131-140


References


- Camenisch, Shoup: Practical Verifiable Encryption and Decryption of Discrete Logarithms. CRYPTO 2003: 126-144


- D. Chaum: *Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms*. In Communications of the ACM.


- T. ElGamal: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In Advances in Cryptology - CRYPTO '84.
Zero Knowledge Proofs: Security

**Proof of Knowledge Property:**

If prover is successful with non-negligible probability, then she “knows” \( x = \log_g y \), i.e., ones can extract \( x \) from her.

Assume \( c \in \{0,1\}^k \) and consider execution tree:

If success probability for any prover (including malicious ones) is \( > 2^{-k} \) then there are two *accepting* tuples \((t,c1,s1)\) and \((t,c2,s2)\) for the same \( t \).
Zero Knowledge Proofs: Security

Prover might do protocol computation in any way it wants & we cannot analyse code.

Thought experiment:

- Assume we have prover as a black box $\rightarrow$ we can reset and rerun prover
- Need to show how secret can be extracted via protocol interface

$$ t = g^s y^c = g^{s'} y^{c'} $$ $\rightarrow$ $$ y^{c'-c} = g^{s-s'} $$ $\rightarrow$ $$ y = g^{(s-s')/(c'-c)} $$ $\rightarrow$ $$ x = (s-s')/(c'-c) \mod q $$
Zero-Knowledge property:

If verifier does not learn anything (except the fact that Alice knows $x = \log g y$)

Idea: One can simulate whatever Bob “sees”.

Choose random $c', s'$
compute $t := g^{s'} y^{c'}$

if $c = c'$ send $s' = s$, otherwise restart

Problem: if domain of $c$ too large, success probability becomes too small
One way to modify protocol to get large domain $c$:

Prover:
- Random $r$
- $t := g^r$
- $h := H(c, v)$
- $s := r - cx$

Verifier:
- Random $c, v$
- $h := H(c, v)$
- $t = g^s y^c$?

notation: $PK\{ (\alpha) : y = g^\alpha \}$
Zero Knowledge Proofs: Security

One way to modify protocol to get large domain $c$:

Choose random $c'$, $s'$
compute $t' := g^{s'} y^{c'}$

after having received $c$ “reboot” verifier

Choose random $s$
compute $t := g^s y^c$
send $s$
Let \( g, h, C_1, C_2, C_3 \) be group elements.

Now, what does
\[
\text{PK\{ (} \alpha_1, \beta_1, \alpha_2, \beta_2, \alpha_3, \beta_3 \text{): } C_1 = g^{\alpha_1} h^{\beta_1} \land C_2 = g^{\alpha_2} h^{\beta_2} \land C_3 = g^{\alpha_3} h^{\beta_3} \land C_3 = g^{\alpha_1} g^{\alpha_2} h^{\beta_3} \}
\]
mean?

→ Prover knows values \( \alpha_1, \beta_1, \alpha_2, \beta_2, \beta_3 \) such that
\[
C_1 = g^{\alpha_1} h^{\beta_1}, \quad C_2 = g^{\alpha_2} h^{\beta_2} \quad \text{and}

C_3 = g^{\alpha_1} g^{\alpha_2} h^{\beta_3} = g^{\alpha_1 + \alpha_2} h^{\beta_3} = g^{\alpha_3} h^{\beta_3}
\]
\[
\alpha_3 = \alpha_1 + \alpha_2 \pmod{q}
\]

And what about:
\[
\text{PK\{ (} \alpha_1, \ldots, \beta_3 \text{): } C_1 = g^{\alpha_1} h^{\beta_1} \land C_2 = g^{\alpha_2} h^{\beta_2} \land C_3 = g^{\alpha_3} h^{\beta_3} \land C_3 = g^{\alpha_1} (g^5)^{\alpha_2} h^{\beta_3} \}
\]

→
\[
C_3 = g^{\alpha_1} g^{\alpha_2} h^{\beta_3} = g^{\alpha_1 + 5 \alpha_2} h^{\beta_3}
\]
\[
\alpha_3 = \alpha_1 + 5 \alpha_2 \pmod{q}
\]
Some Example Proofs and Their Analysis

Let $g, h, C_1, C_2, C_3$ be group elements.

Now, what does

$$\text{PK}\{ (\alpha_1, \ldots, \beta_3) : \quad C_1 = g^{\alpha_1} h^{\beta_1} \land C_2 = g^{\alpha_2} h^{\beta_2} \land C_3 = g^{\alpha_3} h^{\beta_3} \land C_3 = C_2^{\alpha_1} h^{\beta_3} \}$$

mean?

$\rightarrow$ Prover knows values $\alpha_1, \beta_1, \alpha_2, \beta_2, \beta_3$ such that

$C_1 = g^{\alpha_1} h^{\beta_1}$, $C_2 = g^{\alpha_2} h^{\beta_2}$ and

$C_3 = C_2^{\alpha_1} h^{\beta_3} = (g^{\alpha_2} h^{\beta_2})^{\alpha_1} h^{\beta_3} = g^{\alpha_2 \cdot \alpha_1} h^{\beta_3 + \beta_2 \cdot \alpha_1}$

$C_3 = g^{\alpha_2 \cdot \alpha_1} h^{\beta_3 + \beta_2 \cdot \alpha_1} = g^{\alpha_3} h^{\beta_3'}$

$a_3 = a_1 \cdot a_2 \pmod{q}$

And what about

$$\text{PK}\{ (\alpha_1, \beta_1, \beta_2) : \quad C_1 = g^{\alpha_1} h^{\beta_1} \land C_2 = g^{\alpha_2} h^{\beta_2} \land C_2 = C_1^{\alpha_1} h^{\beta_2} \}$$

$\rightarrow$ $a_2 = a_1^2 \pmod{q}$
Let \( g, h, C_1, C_2, C_3 \) be group elements.

Now, what does
\[
\text{PK}\{\alpha_1, \ldots, \beta_2\} : \quad C_1 = g^{a_1} h^{\beta_1} \land C_2 = g^{a_2} h^{\beta_2} \land g = (C_2/C_1)^{a_1} h^{\beta_2}
\]
mean?

→ Prover knows values \( \alpha, \beta_1, \beta_2 \) such that

\[
C_1 = g^{a_1} h^{\beta_1}
\]

\[
g = (C_2/C_1)^{a_1} h^{\beta_2} = (C_2 g^{-a_1} h^{-\beta_1})^{a_1} h^{\beta_2}
\]

→

\[
g^{1/a_1} = C_2 g^{-a_1} h^{-\beta_1} h^{\beta_2/a_1}
\]

\[
C_2 = g^{a_1} h^{\beta_1} h^{-\beta_2/a_1} g^{1/a_1} = g^{a_1 + 1/a_1} h^{\beta_1-\beta_2/a_1}
\]

\[
C_2 = g^{a_2} h^{\beta_2}
\]

\[
a_2 = a_1 + a_1^{-1} \pmod{q}
\]
Pedersen's Scheme:

Choose \( r \in \mathbb{Z}_q \) and compute \( c = g^x h^r \)

Perfectly hiding:
Let \( c \) be a commitment and \( u = \log_g h \)

Thus \( c = g^x h^r = g^{x+ur} = g^{(x+ur')+(r-r')} \)

\[ = g^{x+ur'} h^{r-r'} \quad \text{for any } r'! \]

I.e., given \( c \) and \( x' \) here exist \( r' \) such that \( c = g^{x'} h^{r'} \)

Computationally binding:
Let \( c, (x', r') \) and \( (x, r) \) s.t. \( c = g^{x'} h^{r'} = g^x h^r \)

Then \( g^{x'-x} = h^{r-r'} \) and \( u = \log_g h = (x'-x)/(r-r') \mod q \)