RSAConference2018

San Francisco | April 16 – 20 | Moscone Center

NOW MATTERS

SESSION ID: PDAC-F01

TWO KEYS ARE BETTER THAN ONE BUT THREE KEYS ARE BETTER THAN TWO

Phillip Hallam-Baker

Vice President and Principal Scientist
Comodo Group Inc.

Context:

I don't work for a Certificate Authority.

1 Key Cryptography

=

Good

2 Key Cryptography

=

Better

3 Key Cryptography

=

?

Why are 2 keys better than 1?

- Each key is used to perform a different role
  - Encryption key is not the Decryption key
  - Verification key is not the Signature key

- Public key cryptography is really separation of roles
  - Alice can do one thing
  - Bob can do another
  - What about Carol?

Three (or more) Key cryptography is not new

- Secret sharing [Shamir 1979, Blakely 1979]
- Proxy Re-Encryption [Matt Blaze et. al. 1998]
- Distributed Key Generation [Torben Pedersen 1191]

- Amazing stuff. Why aren't we using it?

Plan of this talk

- Show 3 (and more) Key cryptography is needed
- Introduce building blocks
- Show how to solve practical problems using building blocks
- Demonstrate Open Specification applying multi-key techniques

It’s all about data at rest

- Manning – Data at rest in cable database
- Snowden – Data at rest on NSA server
- DNC – Data at rest on server
- Equifax – Data at rest on server
- Adult Friend Finder – Data at rest on server
- Yahoo – Data at rest on server

- Every one of the 17 largest breaches listed by CSO is Data at Rest

Conclusion

Protect your data at rest

The Insideous Problem

Encrypted Data

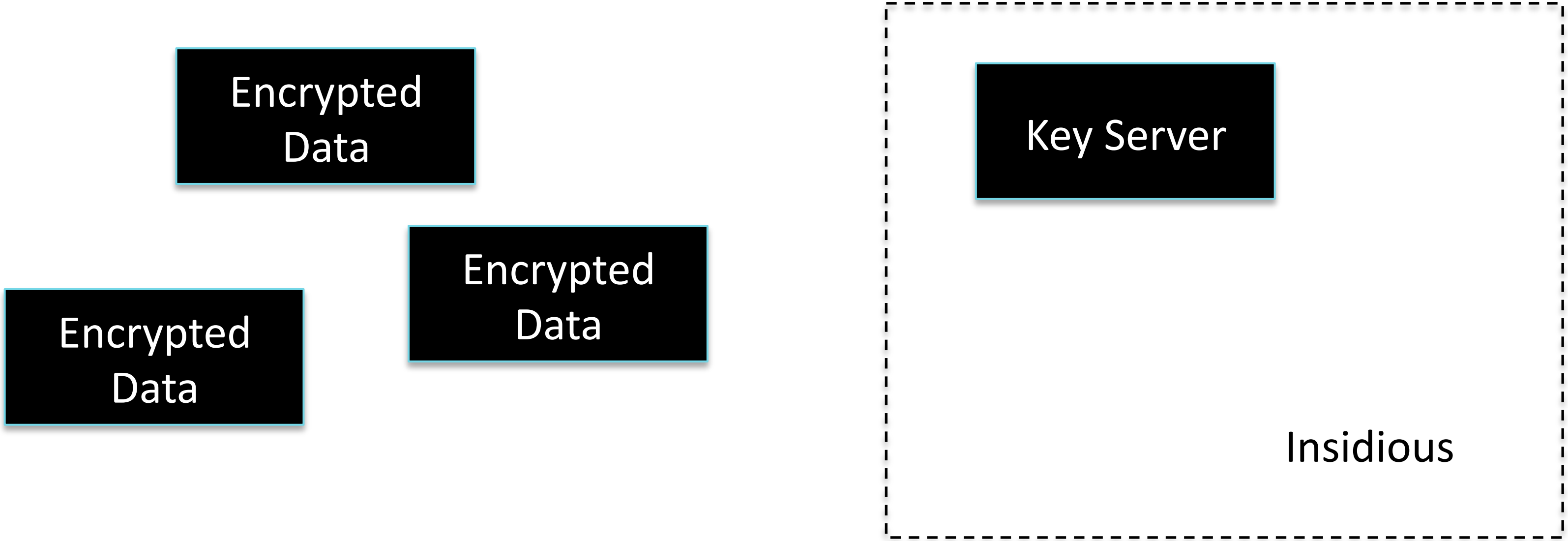
Encrypted Data

Encrypted Data

Key Server

Insidious

Defeating Insideous



Extended Diffie Hellman

- For any DH crypto scheme
  - Let {X, x}, {Y, y}, {E, e} be {public, private} key pairs
- Key Combination Law
  - We can create a new key {Z, z} where $Z = X \otimes Y$ and $z = x + y$
- Result Combination Law
  - $(x \odot E) \oplus (y \odot E) = (z \odot E) = (e \odot Z)$

Diffie Hellman in Discrete Log

- Public Key: $X = g^x \bmod p$
  - $Y = g^y \bmod p$, $Z = g^z \bmod p$

- Key Combination law: $Z = X \otimes Y$ and $z = x + y$
  - $X \otimes Y = X.Y \bmod p$

    $= ((g^x \bmod p) \,.\, (g^y \bmod p)) \bmod p$

    $= g^{(x+y)} \bmod p$

    $= g^z \bmod p$

    $= Z$

    Q.E.D.

Result Combination Law

- $z = x+y$, $Z = g^z \bmod p$, $E = g^e \bmod p$
  - $z \odot \mathrm{E} = e \odot Z = g^{ez} \bmod p$
- Result combination law A $\oplus$ B = A.B *mod p*
  - $(\mathrm{x} \odot \mathrm{E}) \oplus (\mathrm{y} \odot \mathrm{E}) = (g^{ex} \bmod p \,.\, g^{ey} \bmod p) \bmod p$

    $= g^{e(x+y)} \bmod p$

    $= g^{ez} \bmod p$

    $= z \odot \mathrm{E} = e \odot Z$

    Q.E.D.

Elliptic Curve Diffie Hellman

- Public Key: $X = x.Q$
  - $Y = y.Q, Z = z.Q$
- Key Combination law: {Z, z} where Z = X $\otimes$ Y and z = x + y
  - $X \otimes Y = X+Y$

    $= x.Q + y.Q$

    $= (x+y).Q$

    $= z.Q$

    $= Z$

    Q.E.D.

Result Combination Law

- $z = x+y$, $Z = z.Q$, $E = e.Q$
  - $z \odot \mathrm{E} = e \odot Z = \mathrm{z.E} = \mathrm{e.Z} = ez.Q$
- Result combination law $A \oplus B = A + B$
  - $(\mathrm{x} \odot \mathrm{E}) \oplus (\mathrm{y} \odot \mathrm{E}) = x.E + y.E$

    $= (x+y).E$

    $= z.E$

    $= z \odot \mathrm{E} = e \odot Z$

    Q.E.D.

Digression

- Key addition law works for any Elliptic Curve flavor
  - (They are all isomorphic)
- But, some forms are easier to use than others
  - Libraries for Montgomery curves typically lack arbitrary point addition
- IETF picked the 'hard' curves
  - Curve 25519 and Curve 448 [RFC 7748] are recommended for encryption
  - Curve Ed25519 and Ed448 recommended for signature work better (!)

APPLICATION

Distributed Key 'Generation'

- Private Key A
  - Is generated in trusted computing module and bound to the machine
- Private Key B
  - Is generated by application code instance

- Result:
  - Application code will only work on one machine
  - Side channel leaks of application level key do not compromise composite
  - See: MELTDOWN, SPECTRE

Cooperative Key Generation

- What Distributed Key 'Generation' sounds like
  - Generate two key parts independently
  - Combine to form the composite key

- Alice generates seed s_A, uses $H(s_A)$ as private key, sends **public** to CA
- CA generates seed s_C, uses $H(s_C)$ as private key, sends **private** to Alice
- CA generates certificate for composite public key

- Composite private key is at least as random as the two contributions
- CA does not know the private key and cannot guess if Alice key is strong

Signal Key Exchange

- Uses keys as nonces to achieve Perfect Forward Secrecy
- Traditional DH exchange
  - Alice: Publishes g^a Calculates a, $g^b \rightarrow g^{ab}$
  - Bob: Publishes g^b Calculates b, $g^a \rightarrow g^{ab}$

- Signal [pre-key bundle, eliding details…]
  - Alice: Publishes g^a , g^e Calculates a, e, $g^b \rightarrow g^{ab}$, $g^{eb} \rightarrow H(g^{ab}, g^{eb} \ldots)$
  - Bob: Publishes g^b Calculates b, g^a, $g^e \rightarrow g^{ab}$, $g^{eb} \rightarrow H(g^{ab}, g^{eb} \ldots)$

Alternative Approach

- Alice: Private *a*, Generates unique *x* as per message nonce

  Publishes *A, X*

  Calculates *a, x, B, Y* $\rightarrow (a+x) \odot (B \otimes Y)$

- Bob: Private *b,* Generates unique *y* as per message nonce

  Publishes *B, Y*

  Calculates *b, y, A, X* $\rightarrow (b+y) \odot (A \otimes X)$

  $= (a+x) \odot (B \otimes Y)$

- *Proof of security of exchange flows directly from DH proof of security*

Traditional CRM = rebadged DRM

Encrypted Data

Encrypted Data

Encrypted Data

Key Server

Insidious

(+ marketing)

Recryption Approach

- Key Server authorization is necessary but not sufficient.
- Administrator creates Group Encryption Key
  - Private key is Group Administration Key a
  - Public key is Group Encryption Key $A = g^a \bmod p$

- Data Encryption is unchanged [El Gamal Encryption]
  - Bob generates new random private key n
  - Key exchange value is $N = g^n \bmod p$
  - Data is encrypted under KDF ($g^{an} \bmod p$)

Decryption

- The Administration key is split into two parts
  - $a = r + d$
  - r = per user recryption key
  - d = per user decryption key
- Each member added to the recryption group has a different *r, d*

Protocol

- The decryption key *d* is sent to the user
- The recryption key $r = a-d$ is sent to the key server

- Neither the user nor the key server can decrypt without the other

- $g^{ad} \,.\, g^{ar} = g^{a(d+r)} = g^{an}$
  - $(d \odot A) \oplus (r \odot A) = (d+r) \odot A = z \odot A$

Optimization

- A user will typically belong to multiple recryption groups
  - Key management issue

- Simplification
  - Each user generates one encryption/decryption keypair for recryption use
  - Administrator encrypts user’s group decryption key under user encryption key
  - Encrypted decryption key is stored on key server

Why don't we use this today?

- Original algorithm did not meet full requirements
  - Data encrypted under public key X is re-encrypted to public key Y
  - In the Mesh/Recrypt protocol, the administrator knows every private key

  - This does not matter
    - Keys are cheap
    - The administrator can decrypt ALL information encrypted to the group

  - Why did this become such an obstacle?
    - We asked the wrong question

Applications

- Provide cryptographic enforcement of file system ACLs
- Enable end-to-end secure Web
- Enable end-to-end secure Mailing lists, group chat, etc.

- Fast:
  - Decryption requires 1 UDP round trip + 3 private key operations

DEMONSTRATION

Alice Creates a Recryption Group

Alice's Laptop

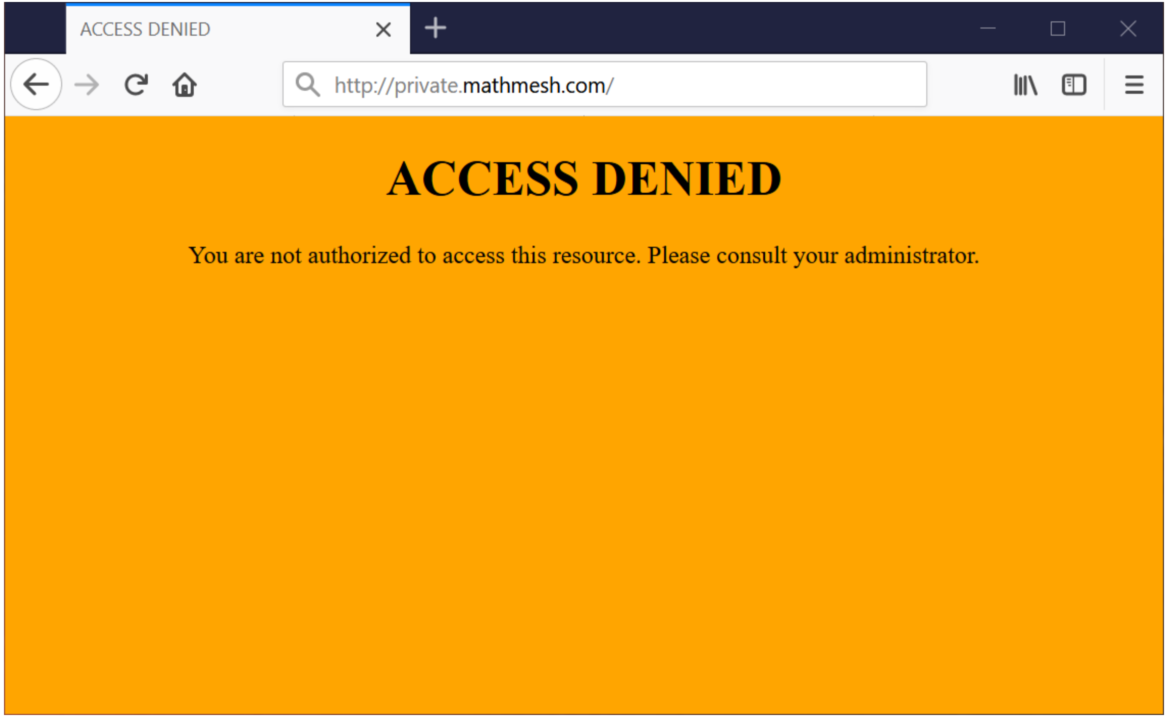
```
$ dareman register alice@cryptomesh.org
Created new personal profile MDLPI-SKSON-LANSR-5I5T3-SVIKR-XSQQF
Profile registered to alice@cryptomesh.org
$ dareman group create authorized@cryptomesh.org
Created
```

Wanda Encrypts a Resource

Wanda's Desktop

```
$ dareman encrypt index.html authorized@cryptomesh.org
Files encryted: 1
```

Bob tries to access

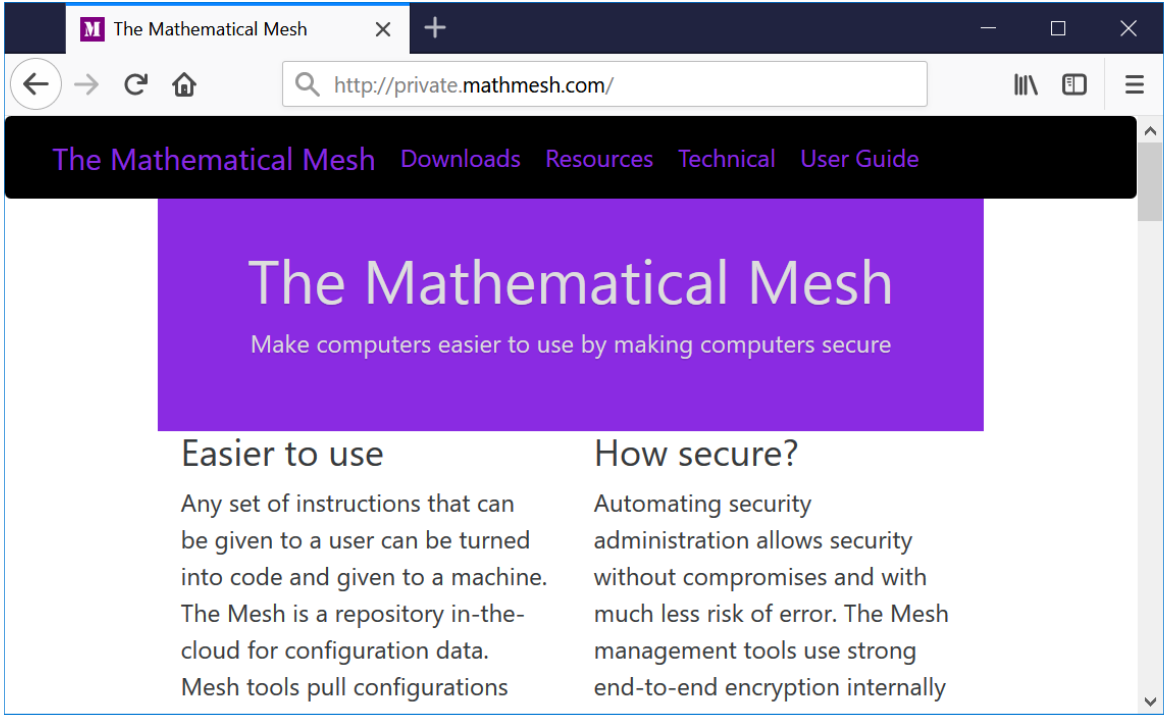


Alice Adds Bob to the Recryption Group

Alice's Laptop

```
$ dareman group add authorized@cryptomesh.org
Created group authorized@cryptomesh.org
$ dareman group add authorized@cryptomesh.org bob@mathmesh.com
Added
```

Bob tries again



Apply – Security Advisors

- Immediate: Make data level security part of your *security policy*
  - Encrypt data at rest
  - Encrypt data as soon as possible
  - Decrypt data as late as possible
- Immediate:
  - Demand data level security in *all* IT RFPs
- Future:
  - Select IT products that meet data level security requirements.
  - Develop IT products that protect data at rest

Apply: Designing multi-key protocols

- Identify the set of roles
  - Each independent role will require a separate key
  - Each Perfect Forward Secrecy enhancement requires an additional key
- Apply the tools
  - Consider applying the DH Key addition rule
  - Consider applying the DH Result combination rule