

# F3ildCrypt: End-to-End Protection of Sensitive Information in Web Services

Matthew Burnside and Angelos D. Keromytis

Department of Computer Science  
Columbia University  
{mb, angelos}@cs.columbia.edu

ISC 2009

# Motivation

- Identity-related information is valuable
- You must provide such information when using an online merchant
- This information is vulnerable to disclosure at the endpoints and in transit
- Can we protect this information end-to-end without revealing details of the logical corporate architecture?

# Outline

- 1 Introduction
- 2 Related work
- 3 Architecture
- 4 Evaluation
- 5 Conclusion

# Introduction

# Merchant trust

Users have to trust online merchants:

- Merchant is not malicious
- Merchant will protect sensitive information for its lifetime
- Merchant site is maintained by diligent sysadmins

# Service-oriented architecture

In this work, we focus on SOAs:

- Requests to a network have a single entry point
- A parent SOA may make requests on multiple child SOAs
- SOAs may operate under differing legal and corporate policies toward private data

In Service Oriented Architectures, users have to trust:

- Merchant *and peer SOAs* are not malicious
- Merchant *and peer SOAs* will protect sensitive information for its lifetime
- Merchant *and peer SOAs* are maintained by diligent sysadmins

# Data in transit

We consider only data in transit across the SOA pipeline

- We protect the data between the web browser and the back-end database
- Our approach does not protect against nodes with legitimate access to the data



# Design alternative

## Pair-wise key distribution

- Generate a public key for each potential destination host in the SOA pipeline
- Deliver public keys to each web browser
- Browser encrypts each field direct to its destination host

# Design alternative (cont.)

## Issues with pair-wise key distribution

- Public keys for all hosts in any partner SOAs must also be delivered
- Public keys must be updated each time the architecture of the SOA or its partners varies
- Reveals the logical architecture of the SOA and its SOA partners

## Related work

# Proxy re-encryption

- Given plaintext  $P$ , Alice  $\langle pk_A, sk_A \rangle$ , and Bob  $\langle pk_B, sk_B \rangle$
- There exists some  $rk_{A \rightarrow B} = F(sk_A, pk_B)$  such that:

$$pk_B(P) = rk_{A \rightarrow B}(pk_A(P))$$

- [Blaze et al., 1998]

Introduced end-to-end encryption in web pipelines

- *“Encryption as a stylesheet”*
- Firefox plugin for application-level crypto
- Requires disclosure of corporate network details
- [Stavrou et al., 2006]

# Architecture

# Architecture

- Network model
- Design goals
- F3ieldCrypt architecture

# Network model

- SOA-style network
- Each SOA may have multiple partner SOAs
- SOAs wish to prevent disclosure of logical architecture and peering



# Design goals

- End-to-end protection of fields – even across SOA boundaries
- Confidentiality of logical architecture of each SOA must be respected

*This work does not focus on providing protection against compromise or failure of entities with legitimate access to sensitive information.*

# F3ieldCrypt architecture

- Each SOA  $s$  publishes a public key  $pk_{E_s}$
- Browser  $b$  generates plaintext  $P$
- $b$  sends  $C = pk_{E_s}(P)$  to  $s$
- Gateway at  $s$  re-encrypts  $C$  to internal hosts and partner SOAs  $l_0 \dots l_n$

# Key generation

- Key pair  $\langle pk_{E_s}, sk_{E_s} \rangle$  generated at the **external-key holder**
- $sk_{E_s}$  used in conjunction with internal application keys  $pk_{I_0} \dots pk_{I_n}$  to generate  $rk_{E \rightarrow I_0} \dots rk_{E \rightarrow I_n}$

# External-key holder

External-key holder has  $sk_{E_s}$  and  $pk_{I_0} \dots pk_{I_n}$  — its compromise could be dangerous. However:

- Very low bandwidth requirements
- Only needed at setup and when adding new internal hosts
- Can be kept offline!

# Key distribution

- $pk_{E_s}$  is publicized
- $rk_{E \rightarrow I_0} \dots rk_{E \rightarrow I_n}$  transmitted to proxy re-encryption engine
- By proxy re-encryption:

$$pk_{I_j}(P) = rk_{E \rightarrow I_j}(pk_E(P))$$

# Proxy re-encryption engine

- Fields arrive at proxy re-encryption engine encrypted under  $pk_{E_s}$
- Each field  $f$  is re-encrypted to  $pk_{I_j}$
- The mapping  $f \rightarrow j$  is determined by an admin-defined XACML policy

# Browser engine

- **Browser** generates plaintext  $P$  containing a set of fields  $f_0...f_n$
- $f_0...f_n$  are encrypted under  $pk_{E_s}$  and delivered to  $s$

# Architecture summary

browser  $\rightarrow$  proxy re-encryption engine:  $pk_{E_s}(f_i)$   
proxy re-encryption engine:  $pk_{I_j}(f_i) = rk_{E \rightarrow I_j}(pk_{E_s}(f_i))$   
proxy re-encryption engine  $\rightarrow$  app  $j$ :  $pk_{I_j}(f_i)$



# Evaluation

# Implementation

- Java-based re-crypto engine uses JHU-MIT Proxy Re-cryptography Library for each browser
- XML gateway at the SOA stores the re-encryption engine
- Python-based XML proxy for each internal application to store keys and unwrap XML

# Testbed servers

## Dell PowerEdge 2650 Servers

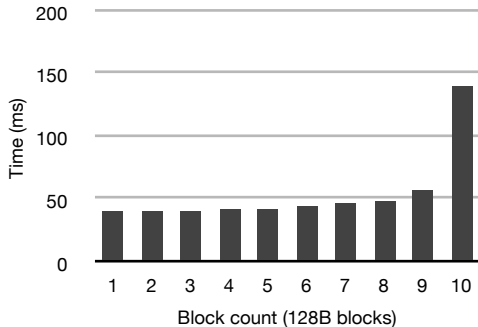
- 2.0GHz Intel Zeon processor, 1GB RAM, Gigabit Ethernet
- OpenBSD 4.2
- OpenBSD PF firewall, Apache 1.3.29, PHP 4.4.1, MySQL 5.0.45

# Testbed client

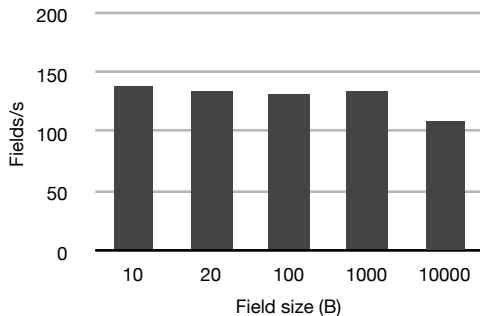
## Macbook Pro

- 2.4 GHz Intel Core 2 Duo, 2GB RAM, Gigabit Ethernet
- OS X 10.5.2, Darwin kernel 9.2.2, Mozilla Firefox 2.0.0.13

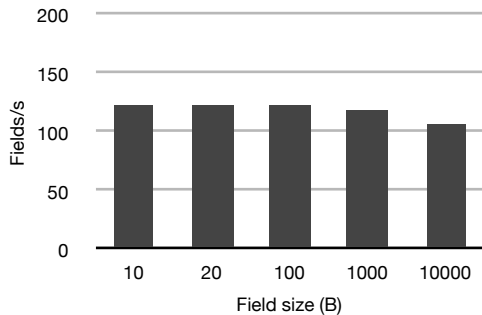
# Block encryption on the client



# Re-encryption rate at an XML gateway



# Decryption rate at an XML proxy





# Conclusion

- End-to-end protection to users
- Protection of logical architecture and partnering for SOAs



# References

-  Matt Blaze, G. Bleumer, and M. Strauss.  
Divertible protocols and atomic proxy cryptography.  
*In Proceedings of Eurocrypt '98*, pages 127–144, 1998.
-  Angelos Stavrou, Michael Locasto, and Angelos Keromytis.  
W3bcrypt: Encryption as a stylesheet.  
*In Proceedings of the 4th Applied Cryptography and Network Security Conference (ACNS 2006)*, pages 349–364, 2006.