# Towards a Provably Secure DoS-Resilient Key Exchange Protocol with PFS[1]

L. Kuppusamy[*†] J. Rangasamy[*†] D. Stebila[*] C. Boyd[*]
J.M. González Nieto[*]

[*]Information Security Institute
Queensland University of Technology, Brisbane, Australia

[†]Society for Electronic Transactions and Security
Chennai, India

IndoCrypt 2011

[1]This work was supported by the Australia-India Strategic Research Fund project TA020002.

# Outline

1. **Introduction**
   - Denial-of-service in Key Establishment
   - Just Fast Keying
2. **Contributions**
   - BPV-JFK
   - DoS-BPV-JFK
3. **Conclusion**

# Key Establishment Protocols

## Goals

Use cryptographic techniques to

- Authenticate each other
- Share a secret key

## Limitations

Involve computationally expensive operations such as modular exponentiation

- This make the server to set a limit on the number of connections at a time
- Vulnerable to a denial-of-service attack

## What is DoS?

- Denial-of-service (DoS) is one of the most common real world network security attacks.
- DoS prevents users from accessing their legitimate resources. It is an attack on *availability*.
- Highly publicised attacks have affected nation states: Estonia (April 2007); Georgia (August 2008); United States and South Korea (July 2009).
- DoS attacks against sites of your choice are readily available for hire.
- Google (June 2009): News searches sparked by Michael Jackson's death were initially mistaken for an automated denial of service attack.

# Types of DoS attacks

- Brute force attacks: attacker generates sufficiently many legitimate-looking requests to overload a server's resources. Does not require special knowledge of protocol specification or implementation.
- Semantic attacks: attacker tries to exploit vulnerabilities of particular network protocols or applications. Requires special knowledge of protocol specification and implementation.

# Two party DoS-resilient key exchange protocols

- Just Fast Keying (JFK)
- Client Aided-RSA (CA-RSA)
- Modified Internet Key Exchange (MIKE)
- Host Identity Protocol (HIP)

# Just Fast Keying (JFK)

W. Aiello, S. M. Bellovin, M. Blaze, R. Canetti, J. Ioannidis, A. D. Keromytis, and O. Reingold.
Just Fast Keying: Key agreement in a hostile Internet.
*ACM Transactions on Information and System Security*, 7(2):1–30, May 2004.

- a simple, efficient and secure key exchange protocol
- well known for its DoS resistant techniques such as re-use of Diffie-Hellman (DH) ephemeral keys
- achieves only adaptive forward secrecy due to the re-use technique
- claimed secure in the CK01 model under the Decisional Diffie-Hellman assumption

# JFK protocol

| Client | | Server |
|---|---|---|
| Nonce $N_c$ | $H(N_c), g^x \longrightarrow$ | |
| | $\longleftarrow g^y, N_s, H(N_c),$ | |
| $K_e, K_a, S_1$ | $N_c, E_c, A_c \longrightarrow$ | verify $A_c$, Decrypt $E_c$ |
| | $\longleftarrow S_2, E_s, A_s,$ | Verify $S_1$, generate $S_2$ |

$$K_e = H_{g^{xy}}(N_s, H(N_c), 1),\ K_a = H_{g^{xy}}(N_s, H(N_c), 2)$$
$$SIG : S_1 = \{s_{k_c}(H(N_c), N_s, g^x, g^y), ID_C\}$$
$$Encryption : E_c = \{S_1\}_{K_e},\ MAC : A_c = \{E_c\}_{K_a}$$
$$S_2 = s_{k_s}(H(N_c), N_s, g^x, g^y, ID_C),\ E_s = \{S_2\}_{K_e},\ A_c = \{E_s\}_{K_a}$$

# Cost-based Analysis of JFK

Smith et al analysed JFK using Meadows Cost-based framework and found two computational based DoS attacks

## An Overview of Meadows cost-based framework

- proposed to analyse DoS Vulnerabilities in network protocols
- Assigns cost to every action of the Client and server
- Calculate the total cost for each party in a specific run of the protocol
- If the total cost of the server (to send a response)is greater than the total cost (to send a message), then the protocol is vulnerable to a DoS attack

# Smith et al's attacks on JFK

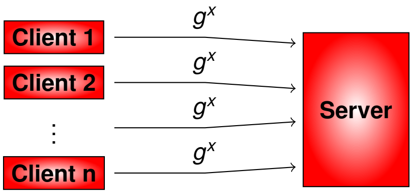
| Client | | Server |
|---|---|---|
| Nonce $N_c$ | $H(N_c), g^x$ → | |
| | ← $g^y, N_s, H(N_c),$ | |
| $K_e, K_a, S_1$ | $N_c, E_c, A_c$ → | verify $A_c$, Decrypt $E_c$ |
| | ← $S_2, E_s, A_s,$ | Verify $S_1$, generate $S_2$ |

$$K_e = H_{g^{xy}}(N_s, H(N_c), 1),\ K_a = H_{g^{xy}}(N_s, H(N_c), 2)$$

## Attack 1

- by a direct application of Meadows framework
- goal is to force the server to perform MAC ($A_c$) verification
- due to the expensive $K_a$ operation
- fix: to incorporate client puzzles

# Smith et al's attack contd.



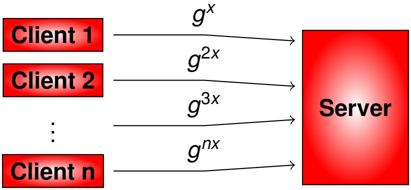

## Attack 2

- possible due to the presence of co-ordinated initiators
- possible when both clients and server re-use $g^x$ and $g^y$
- goal is to force the server to perform sig $S_1$ verification
- Idea: $g^{xy}$ can be amortised across all sessions
- fix: binding the ephemeral keys to a specific session. for example, set the shared DH exponential as $g^{xyr}$, where $r$ is a function of session specific parameters

# Contributions

- A new DoS vulnerability in JFK
- Security flaw: Basic JFK with re-use technique may require GDH assmption not the DDH assumption
- Modified JFK protocol using BPV technique
  - secure under the DDH assumption
  - achieves perfect forward secrecy
- Analysed in Stebila et al model for Dos resilience

# New DoS vulnerability




- possible due to the presence of co-ordinated initiators
- possible when only the server re-use the DH ephemeral keys
- Idea: the malicious client computes ephemeral DH key $g^x$ for one session and then computes other ephemeral DH keys as $g^{nx}$, where $n = 2, 3, ...$. Similar idea is applicable to the computation of the shared DH exponentials ($g^{nxy}$).

## BPV Generator (Boyko, Peinado, Venkatesan Eurocrypt'98)

- Method for computing DH exponential in few multiplications.

### BPV Generator

Let $p$ be a DSA modulus such that the prime $q$ divides $p - 1$. Select a random element $g$ of order $q$ in the multiplicative group $\mathbb{Z}_p^*$. Let $N$ and $\ell$ be integer parameters such that $N \geq \ell \geq 1$.

- **Pre-processing** run once. Generate $N$ random integers $x_1, x_2, \ldots x_N \in \mathbb{Z}_q$. Compute $X_i = g^{x_i} \mod p$ for each $i$ and store the pair $(x_i, X_i)$ in a table.
- **Whenever a pair** $(y, g^y)$ **is needed**: Generate a random set $S \subseteq_R \{1, \ldots, N\}$ such that $|S| = \ell$. Compute $y = \sum_{j \in S} x_j \mod q$. If $y = 0$, stop and generate $S$ again. Otherwise compute $g^y = \prod_{j \in S} g^{x_j} \mod p$ and return $(y, g^y)$.

# Statistical indistinguishability of BPV generator

## Nguyen etal

Let $q$ be a prime, and let $N \geq \ell \geq 1$. Then,

$$\frac{1}{q^N} \sum_{\vec{x} \in \mathbb{Z}_q^N} \sum_{y \in \mathbb{Z}_q} \left| \Pr_{S \subseteq [1,N] : |S| = \ell} \left( \sum_{j \in S} x_j \equiv y \mod q \right) - \frac{1}{q} \right| \leq \sqrt{q / \binom{N}{\ell}}$$

- for appropriate choices of the $N$ and $\ell$ values, the BPV generator outputs almost all the elements of $\mathbb{Z}_q$ and the proportion of elements not output by the BPV generator is very small
- the result holds regardless of whether the pre-computed $x_i$'s are known to a distinguisher or not

# Efficiency

- choose a bigger value of $N$ (polynomial in $\log q$) to make $\ell$ smaller.

| $N$ | $\ell$ | $\sqrt{q/\binom{N}{\ell}}$ | Runtime BPV-Pre (s) | Runtime BPV-Gen (ms) |
|---|---|---|---|---|
| $2^{11} = 2048$ | 48 | $2^{-82}$ | 0.939 | 0.226 |
| $2^{12} = 4096$ | 40 | $2^{-80}$ | 1.892 | 0.196 |
| $2^{13} = 8192$ | 35 | $2^{-81}$ | 3.758 | 0.168 |
| $2^{14} = 16384$ | 31 | $2^{-81}$ | 7.527 | 0.156 |
| $2^{16} = 65536$ | 26 | $2^{-83}$ | 30.148 | 0.134 |

- a single 160-bit modular exponentiation takes 0.461 ms.
- The advantage factor of BPV generation over modular exponentiation based on the parameter values listed in Table is between 2 and 3.4.

# BPV-JFK

| Client | | Server |
|---|---|---|
| Nonce $N_c$ | $H(N_c), g^x$ → | |
| | ← $g^y, N_s, H(N_c),$ | $g^y \leftarrow$ BPVPairGen |
| $N_c, K_e, K_a, S_1$ | $E_c, A_c$ → | verify $A_c$, Decrypt $E_c$ |
| | ← $S_2, E_s, A_s,$ | Verify $S_1$, generate $S_2$ |

$$K_e = H_{g^{xy}}(N_s, H(N_c), 1),\ K_a = H_{g^{xy}}(N_s, H(N_c), 2)$$
$$SIG : S_1 = \{s_{k_c}(H(N_c), N_s, g^x, g^y), ID_C\}$$
$$Encryption : E_c = \{S_1\}_{K_e}, MAC : A_c = \{E_c\}_{K_a}$$
$$S_2 = s_{k_s}(H(N_c), N_s, g^x, g^y, ID_C),\ E_s = \{S_2\}_{K_e}, A_c = \{E_s\}_{K_a}$$

- BPV-JFK achieves Perfect Forward Secrecy (PFS)
- BPV-JFK is not fully DoS resilient. DoS-attack is possible if the server send bogus MAC $A_c$ in the third message

# DoS Resistance in BPV-JFK

- Stebila etal gave a generic technique to transform any protocol into a DoS resistant protocol
- The technique uses strongly difficult interactive client puzzles as a DoS countermeasure and message authentication codes (MAC) for integrity of stateless connections.
- The server in the protocol must not perform any expensive operation until it verifies the MAC and the puzzle solution.

# DoS-BPV-JFK

| **Client** | | **Server** |
|---|---|---|
| Nonce $N_c$ | $H(N_c)$, $g^x$ $\longrightarrow$ | |
| | $\longleftarrow$ MAC, CPuz,$g^y$, $N_s$, $H(N_c)$, | $g^y \leftarrow$ BPV pair gen |
| $K_e, K_a, S_1$ | MAC, PuzSoln, $N_c, E_c, A_c$ $\longrightarrow$ | verify MAC, CPuz, $A_c$, Decrypt $E_c$ |
| | $\longleftarrow$ $S_2, E_s, A_s,$ | Verify $S_1$, generate $S_2$ |

$$K_e = H_{g^{xy}}(N_s, H(N_c), 1),\ K_a = H_{g^{xy}}(N_s, H(N_c), 2)$$
$$SIG : S_1 = \{s_{k_c}(H(N_c), N_s, g^x, g^y), ID_C\}$$
$$Encryption : E_c = \{S_1\}_{K_e}, MAC : A_c = \{E_c\}_{K_a}$$
$$S_2 = s_{k_s}(H(N_c), N_s, g^x, g^y, ID_C),\ E_s = \{S_2\}_{K_e},\ A_c = \{E_s\}_{K_a}$$

# Comparison

| Protocol | Cost-based vulnerability | Security assumptions | Perfect Forward Secrecy | DoS-resilience |
|---|---|---|---|---|
| JFK | Yes | GDH, ROM | Only with no reuse | No |
| DoS-JFK | No | GDH, ROM | Only with no reuse | Yes |
| BPV-JFK | No | DDH | Yes | No |
| DoS-BPV-JFK | No | DDH | Yes | Yes |

**Table:** Comparison of properties of JFK-based protocols

## Conclusion

- DoS may arise in a number of ways. Our focus is on resource exhaustion DoS attacks (on network protocols)
- we propose to use a technique introduced by Boyko et al. to achieve PFS and to resist the identified attack on JFK
- BPV-JFK is secure in CK01 model under the DDH assumption
- BPV-JFK is DoS resilient after incorporating client puzzles and secure MACs.

Thank You all