Elliptic Curve Cryptography (ECC) for Host Identity Protocol (HIP)

Oleg Ponomarev, Andrey Khurri, and Andrei Gurtov
Helsinki Institute for Information Technology HIIT /
Aalto University, Espoo, Finland
Email: firstname.lastname@hiit.fi

***Abstract*—We compare computational resources required for handling control plane of the Host Identity Protocol (HIP) using Rivest-Shamir-Adleman (RSA) versus Elliptic Curve Cryptography (ECC) encryption algorithms with keys of equivalent strength. We show that servers would establish almost three times more HIP connections per second when ECC is used for generating the session key. For devices with low computational power such as Nokia N810 Internet Tablet, the use of ECC would notably reduce the delay to establish a HIP association.**

Unless compatibility with legacy RSA/DSA-only systems is needed, the Host Identity may be an ECC key as well, but such a modification would bring only 50 percent additional performance with the current default keys. However the situation becomes different under higher security requirements when employing ECC for the host identification boosts the performance more than four times, and we consider ECC Host Identities desirable in that case.

I. Introduction

Currently IP addresses serve in the Internet as both end host identifiers and routing locators. This principle worked well before the wide adoption of portable devices frequently changing their IP addresses. There are various efforts to split the locator and identifier roles of IP addresses [15], [20]. One of the approaches known as Host Identity Protocol (HIP) was introduced in 1999 by Robert Moskowitz [9]. The architecture [10] and details of the protocol [11] were adopted as Request-for-Comments (RFCs). Currently, there are three active implementations [5], [16], [3] for various operating systems.

The Host Identity Protocol [4] uses cryptographic algorithms to authenticate communicating hosts to each other and establish a secure channel between them. So far only Rivest-Shamir-Adleman (RSA) [21] and Digital Signature Algorithm (DSA) [13] along with Diffie-Hellman (DH) [2] exchange are allowed in the HIP specifications. In this article, we investigate how the introduction of Elliptic Curve Cryptography (ECC) [6], [8] methods would affect performance of servers using HIP and the delay to establish a HIP association by a lightweight client. We give a short overview of the protocol, RSA and ECC in Section II, present the results of our measurements in Section III, estimate protocol performance with various key combinations in Section IV, and conclude the article in the last section.

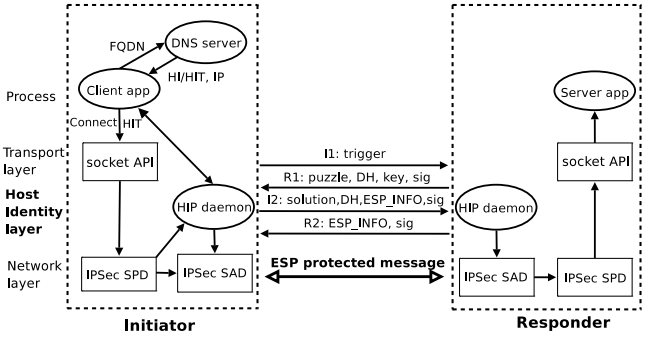

Figure 1. HIP architecture.

II. Background

A. HIP architecture

The idea behind HIP is based on decoupling the network layer from the higher layers in the protocol stack architecture (see Figure 1). HIP defines a new global namespace, the Host Identity namespace, thereby splitting the double meaning of IP addresses. When HIP is used, above layers do not any more rely on IP addresses as host names. Instead, Host Identities are used in the transport protocol headers for establishing connections. IP addresses at the same time act purely as locators and are responsible for routing packets towards the destination. For compatibility with IPv6 legacy applications, a Host Identity is represented by a 128-bit long hash, the Host Identity Tag (HIT).

Figure 1 illustrates the overall HIP architecture including the process of establishing a HIP association called base exchange (BEX) [10]. HIP offers several benefits including end-to-end security, resistance to CPU and memory exhausting denial-of-service (DoS) attacks, NAT traversal, mobility and multihoming support.

B. HIP base exchange

To start communicating through HIP, two entities must perform BEX, which consist of four messages transferred between the initiator and the responder (see Figure 2). The messages are named by letters and numbers. The letters denote the sender of the packet, I for initiator and R for responder. The numbers are simply sequential. Hence, the four messages are named as $I1$, $R1$, $I2$, and $R2$.

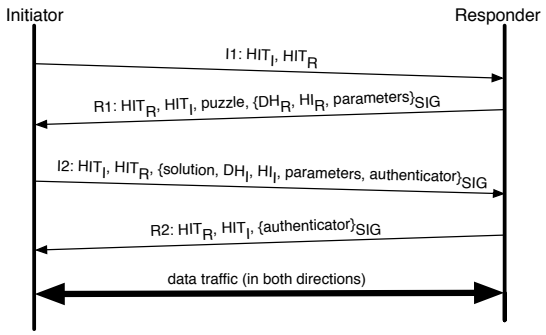

Figure 2. HIP base exchange.

An $I1$ message is a mere trigger. It is used by the initiator to request an $R1$ message from the responder.

An $R1$ message contains a public Diffie-Hellman key and a public Host Identity key of the responder. The Diffie-Hellman key in the $R1$ message allows the initiator to compute the Diffie-Hellman session key. Hence, when constructing an $I2$ message, the initiator already has a session key and can use the keys derived from it.

The $I2$ message is the main message of the protocol. It contains the initiator's public Diffie-Hellman key, the initiator's public Host Identity key, optionally encrypted with the Diffie-Hellman key, and an authenticator showing that the $I2$ message has been recently constructed by the initiator.

Once the responder has received the $I2$ message, it can continue to construct the Diffie-Hellman session key, to decrypt the initiator's Host Identity public key (if encrypted), and to verify the authenticator. If the verification succeeds, the responder is assured that there is a host that has access to the private key corresponding to the initiators Host Identity public key. In addition, both hosts are confident that they share a common Diffie-Hellman session key, which is not known to other hosts. The responder then computes an authenticator and sends it in an $R2$ packet to the initiator.

Another purpose of BEX is to create a pair of IPsec Encapsulated Security Payload (ESP) Security Associations (SAs), one for each direction. All subsequent traffic between communicating parties is protected by IPsec.

The communication channel may be terminated with HIP_CLOSE and HIP_CLOSE_ACK messages, each containing a signature. A summary of the time-consuming operations in HIP control plane is shown in Table I, where *Total* row contains formulae to estimate resources for the complete connection cycle and *BEX Only* row may be used for the delay to establish the connection by a client.

C. RSA and ECC

Since RSA is the default algorithm for HIP [11], it was used as the base for the estimations. RSA was invented by Rivest, Shamir and Adleman in 1977 [21]. Its fundamental operation is modular exponentiation and security of RSA is based on the difficulty of factoring large integers.

MIT was granted a U.S. Patent for "Cryptographic communications system and method" [22] in 1983, but it was released to the public domain by RSA Security on 21 September 2000.

ECC operates on groups of points over elliptic curves. The strength of ECC relies on the complexity of the elliptic curve discrete logarithm problem (ECDLP), which is much harder to solve computationally than with ECC's counterparts. The fundamentals of ECC were published independently by Neal Koblitz [6] and Victor Miller [8] in 1985.

There are sub-exponential algorithms for the integer factorization problem, but only exponential algorithms for the ECDLP [7]. Hence, ECC achieves the same level of security with smaller key sizes as shown in Table II. Although with present requirements to security levels ECC does not bring much efficiency comparing with the "mainstream" algorithms such as DH or RSA, significant benefits are expected from ECC in the future when the threats in communication systems will multiply from their current level [8].

One issue, which substantially differentiates ECC from the rest of cryptography methods and at the same time complicates its wide adoption, comes from numerous ECC patents. A notable patenter is the Canadian company *Certicom*[1] that holds over 100 patents on ECC and public-key cryptography in general [14]. Since the introduction of ECC, the company has been continuously seeking to develop an efficient implementation of ECC, which resulted in a commercial ECC toolkit that can be used with various applications. Since 1997, Certicom organizes the *Certicom ECC Challenge*[2] that gives anyone an opportunity to solve the ECDLP at its current level. Participants of the challenge are supposed to derive the ECC private keys based on the list of the public keys and other known parameters. To date, the highest solved challenges are 109-bit. The 131-bit challenge requires much more resources to be solved whereas higher challenges (starting from 163-bit) are treated as computationally infeasible.

Despite complex IPR issues, several countries around the world are adopting parts of ECC via licensing. For instance, the National Security Agency (NSA) has purchased from Certicom the rights to use a set of ECC techniques under patents. The goal is to ensure that future communication

[1]http://www.certicom.com/index.php/ecc
[2]http://www.certicom.com/index.php/the-certicom-ecc-challenge

Table I
CRYPTOGRAPHIC OPERATIONS DURING BEX.

Message	Initiator	Responder
$I1$	-	-
$R1$	verify, DH_compute_key	sign
$I2$	sign	verify, DH_compute_key
$R2$	verify	sign
CLOSE	sign	verify
CLOSE_ACK	verify	sign
Total	$2 \times T_s + 3 \times T_v + T_{dh}$	$3 \times T_s + 2 \times T_v + T_{dh}$
BEX Only	$T_s + 2 \times T_v + T_{dh}$	$2 \times T_s + T_v + T_{dh}$

Table II
COMPARABLE KEY SIZES (BITS) WITH DIFFERENT CRYPTOSYSTEMS [1].

Security level	**ECC**	**DSA/RSA**
80	160	1024
112	224	2048
128	256	3072
192	384	7680
256	512	15360

systems will be capable of protecting sensitive information for the US government and country [14].

III. TIMING OF CRYPTOGRAPHIC OPERATIONS

When HIP is used, most of the CPU resources are consumed by several cryptographic operations during BEX. In this section we present our measurement results of the operations listed in Table I. First, we describe our measuring methodology and validate it by comparing the results with our previous study. Next, we introduce the results obtained on a Linux server and a Nokia N810 Internet Tablet.

A. Methodology

To measure the processor time spent by different cryptographic operations, we used the OpenSSL library [17] and ran several benchmarks. The OpenSSL speed measurement utility *openssl speed* was used for timing *RSA_sign*, *RSA_verify*, *ECDSA_sign* and *ECDSA_verify* functions. The utility executes an operation in a cycle for about 10 seconds and then uses the function *getrusage* to account the processor time. Our own utility was used to execute *DH_compute_key* and *ECDH_compute_key* for about 90 seconds.

B. Validation

We started with benchmarking OpenSSL version *0.9.8g* used for stress-testing of HIP implementations [19] at the same HP ProLiant DL360 G5 server with two quad-core Intel Xeon E5440 2.83-GHz processors running Linux x86_64 kernel version 2.6.29.6. The measurement results obtained using only one CPU core are shown in Table III. The table presents the average time needed by the server to compute a 1536-bit DH key and to perform sign and verify operations with a 1024-bit RSA key. These key sizes are default now and were used in the stress-test.

The number of BEX per second for the responder (i.e., a HIP server in our case) can be estimated using the *Total Time* formula from Table I. Although the calculation $1000000/(3*778+2*37+6624)$ (performed using the values from Table III) gives us about 111 BEX per second, it corresponds well to the maximum of 112 BEX per second achieved in our previous work [19]. This validates our methodology and allows us using the *Total Time* formula with the separately measured operations' durations to calculate the total BEX time and the maximum number of BEX per second.

Naturally there is always a processing overhead caused by handling network traffic, processing connection parameters and other operations, but profiling and stress-testing results confirm that with the usual key sizes it was not significant. The overhead fraction would grow with very small keys or an extensive control packet retransmission due to bad network connection, but we did not take it into account for the sake of simplicity.

Table III
RESULTS FOR OPENSSL VERSION 0.9.8G.

Operation	Time
DH_compute_key using 1536-bit group	6624 μs
RSA_sign using 1024-bit key	778 μs
RSA_verify using 1024-bit key	37 μs

C. Linux server

We used the same ProLiant DL360 G5 server with two quad-core Intel Xeon E5440 2.83-GHz processors running Linux x86_64 kernel version 2.6.29.6 to measure the duration of cryptographic operations with a newer OpenSSL version *1.0.0-beta2*. The results with various key sizes are shown in Table IV and Table V.

Table IV
SIGNATURE OPERATIONS AT LINUX SERVER.

Key size	Sign	Verify
RSA using 1024-bit key	571 μs	28 μs
RSA using 2048-bit key	3447 μs	100 μs
ECDSA using 160-bit key	91 μs	355 μs
ECDSA using 192-bit key	134 μs	583 μs
ECDSA using 224-bit key	168 μs	769 μs
ECDSA using 256-bit key	224 μs	1080 μs

Table V
DIFFIE-HELLMAN EXCHANGE AT LINUX SERVER.

Operation	Time
DH_compute_key using 768-bit group	0.750 ms
DH_compute_key using 1536-bit group	5.147 ms
DH_compute_key using 2048-bit group	11.836 ms
DH_compute_key using 3072-bit group	38.449 ms
ECDH_compute_key using 160-bit key	0.296 ms
ECDH_compute_key using 192-bit key	0.384 ms
ECDH_compute_key using 224-bit key	0.634 ms
ECDH_compute_key using 256-bit key	0.646 ms

D. Nokia Internet Tablet

The same benchmarks were executed on a Linux-based Nokia N810 Internet Tablet, powered by a 400-MHz ARM CPU and running the Linux kernel version 2.6.21. The benchmarks on the N810 used the OpenSSL version *1.0.0-beta3*. The results of the experiments are presented in Table VI and Table VII.

Table VI
SIGNATURE OPERATIONS AT N810.

Key size	Sign	Verify
RSA using 1024-bit key	24.038 ms	1.373 ms
RSA using 2048-bit key	160.794 ms	5.008 ms
ECDSA using 160-bit key	2.569 ms	9.407 ms
ECDSA using 192-bit key	4.177 ms	18.710 ms
ECDSA using 224-bit key	5.509 ms	26.981 ms
ECDSA using 256-bit key	6.720 ms	33.322 ms

Table VII
DIFFIE-HELLMAN EXCHANGE AT N810.

Operation	Time
DH_compute_key using 768-bit group	33.854 ms
DH_compute_key using 1536-bit group	248.372 ms
DH_compute_key using 2048-bit group	576.563 ms
ECDH_compute_key using 160-bit key	7.670 ms
ECDH_compute_key using 192-bit key	11.833 ms
ECDH_compute_key using 224-bit key	16.252 ms

IV. ESTIMATED BEX PERFORMANCE AND ANALYSIS

The results presented in the previous section allow us to estimate the number of connections that can be established by a server with various key sizes. This section presents such estimations for our Linux server and Nokia N810 Internet Tablet.

The current default combination of a 1024-bit RSA Host Identity for authentication and a 1536-bit group for the DH exchange was the starting point in our estimations. The next step was to replace the DH operation with the Elliptic Curve Diffie-Hellman (ECDH) of equivalent strength (see Table II) that will allow us to keep an RSA Host Identity for compatibility but spend less time to generate a session key if both parties support ECC. Finally, the RSA Host Identity may be replaced with an Elliptic Curve DSA (ECDSA) key, unless compatibility with legacy RSA/DSA-only systems is needed.

Table VIII contains the estimated number of HIP connections per second that our Linux server is able to process in the aforementioned scenarios. We present the estimations for the key lengths that according to NIST guidelines [12] will ensure a sufficient security level in the coming years. In particular, a minimum of 112 bits of security shall be provided from year 2011 onwards that corresponds to 2048 bits for RSA/DSA keys and to 224 bits for ECDH and ECDSA.

As Table VIII shows, the number of new connections processed by the server increases from the current 145 to 425 per second, i.e. almost three times. Using an ECC key as the Host Identity causes further growth to 639 connections per second, but breaks compatibility with RSA/DSA-only systems.

While for the server's performance evaluation we focused on the number of incoming HIP connections, it is less common characteristic for a resource-constrained device that is usually acting as a client. For this reason, for the Nokia N810 Internet Tablet we estimated the duration of BEX, which is an important metric for interactive applications using HIP on such a client. The estimated delays to establish a HIP association by the N810 are presented in Table IX. In this case, the results indicate that the duration of a HIP association establishment is reduced from 275 ms to 39 ms and 33 ms by applying ECDH and ECDSA respectively. The numbers do not include operations to close the HIP connection.

Table VIII
ESTIMATED NUMBER OF BEX PER SECOND BY ONE CORE OF A LINUX SERVER.

Authentication	Session key	BEX per second
RSA1024	DH1536	145
RSA1024	ECDH192	425
ECDSA160	ECDH192	639
RSA2048	DH2048	45
RSA2048	ECDH224	89
ECDSA224	ECDH224	374

Table IX
ESTIMATED BEX DURATION AT N810.

Authentication	Session key	BEX duration
RSA1024	DH1536	275 ms
RSA1024	ECDH192	39 ms
ECDSA160	ECDH192	33 ms
RSA2048	DH2048	747 ms
RSA2048	ECDH224	187 ms
ECDSA224	ECDH224	129 ms

The effect of utilizing ECC for the Host Identity is much greater with longer keys. The use of a 224-bit ECDSA identity allows the server to accept some 374 instead of 89 connections achieved with a 2048-bit RSA identity, i.e. the estimated growth is more than four times (see Table VIII).

With the Nokia N810 Internet Tablet, the delay to establish a HIP association is reduced from 187 to 129 ms, i.e. by 30 percent (see Table IX).

V. CONCLUSIONS

In this paper we measured the processor resources required by the cryptographic operations during the Host Identity Protocol base exchange. Our estimations show that by adopting ECC for the Diffie-Hellman exchange, the number of new connections handled by a server increases by two-three times. The protocol allows the responder to offer the initiator both DH and ECDH, thus compatibility with RSA/DSA-only systems is not broken.

An enhancement in HIP performance is also observed with a lightweight mobile client such as a Nokia N810 Internet Tablet that performs cryptographic operations more efficiently with ECC, thus reducing the total BEX time by 75-85 percent.

ECC allows to improve the performance even more, when we replace not only the DH by the ECDH, but also the RSA by the ECDSA, i.e. when we use ECC keys for the host authentication as well. However it would break compatibility with systems not supporting ECC.

Because it would give only 50 percent more connections per second with the current default key sizes, we consider it reasonable only for private overlays without non-ECC systems.

Under higher security requirements meaning greater key lengths, ECC Host Identity allows four times more connections per second and notably less time to perform base exchange by a low computational power client. We take it as a significant benefit and suggest for a consideration by the community [18].

The results presented in this work are based on measurements of separate cryptographic operations on real devices and subsequent performance estimations of the protocol running these operations. Our main contribution is a practical illustration of the potential behind Elliptic Curve Cryptography used in a concrete application (i.e., the Host Identity Protocol), which is important to both HIP and ECC communities.

REFERENCES

[1] Certicom Research. Standards for efficient cryptography. SEC 1: Elliptic curve cryptography. Working draft. version 1.9, Certicom Research, Aug. 2008.

[2] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[3] HIP for BSD project, 2005. Available at: http://www.hip4inter.net Accessed 20 January 2010.

[4] A. Gurtov. *Host Identity Protocol (HIP): Towards the Secure Mobile Internet*. Wiley and Sons, 2008.

[5] HIP for Linux. [Online] Available at: http://hipl.hiit.fi Accessed 20 January 2010.

[6] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation, 1987*, 48(177):203–209, jan 1987.

[7] A. Menezes. The elliptic curve discrete logarithm problem: State of the art. In *IWSEC '08: Proceedings of the 3rd International Workshop on Security*, pages 218–218, Berlin, Heidelberg, 2008. Springer-Verlag.

[8] V. S. Miller. Use of elliptic curves in cryptography. In *Lecture notes in computer sciences; 218 on Advances in cryptology—CRYPTO 85*, pages 417–426, New York, NY, USA, 1986. Springer-Verlag New York, Inc.

[9] R. Moskowitz. Host identity payload architecture: draft-moskowitz-hip-arch-00, Dec. 1999.

[10] R. Moskowitz and P. Nikander. Host Identity Protocol architecture. IETF RFC 4423, May 2006.

[11] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. Experimental Host Identity Protocol (HIP). IETF RFC 5201, Apr. 2008.

[12] National Institute of Standards and Technology. *NIST SP 800-57, Recommendation for Key Management – Part 1: General (Revised)*. NIST, Mar. 2007. Available at http://csrc.nist.gov/publications/nistpubs /800-57/sp800-57-Part1-revised2_Mar08-2007.pdf Accessed 20 January 2010.

[13] National Institute of Standards and Technology. *FIPS PUB 186-3: Digital Signature Standard (DSS)*. NIST, June 2009. [Online]. Available at http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf. Accessed 25 Jan 2010.

[14] National Security Agency. The Case for Elliptic Curve Cryptography, Jan. 2009. [Online]. Available at: http://www.nsa.gov/business/programs/elliptic_curve.shtml Accessed 25 January 2010.

[15] P. Nikander. Identifier / locator separation: Exploration of the design space (ILSE), Feb. 2007. Work in progress. Expires: August 30, 2007.

[16] OpenHIP website. [Online] Available at: http://www.openhip.org Accessed 25 January 2010.

[17] OpenSSL: The open source toolkit for SSL/TLS. Available at http://www.openssl.org, 2009.

[18] O. Ponomarev. Additional key algorithms for host identity protocol: draft-ponomarev-hip-ecc-00, Oct. 2009. Work in progress.

[19] O. Ponomarev and A. Gurtov. Stress testing of HIP implementations. In *Proc. of Third International Conference on Internet Technologies and Applications (ITA'09)*, Wrexham, North East Wales, UK, Sept. 2009.

[20] B. Quoitin, L. Iannone, C. de Launois, and O. Bonaventure. Evaluating the benefits of the locator/identifier separation. In *MobiArch '07: Proceedings of 2nd ACM/IEEE international workshop on Mobility in the evolving internet architecture*, pages 1–6, New York, NY, USA, 2007. ACM.

[21] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[22] R. L. Rivest, A. Shamir, and L. M. Adleman. Cryptographic communications system and method. *US Patent 4405829*, Sept. 1983.