View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Crossref

Hindawi
Security and Communication Networks
Volume 2017, Article ID 1467614, 7 pages
https://doi.org/10.1155/2017/1467614

WILEY | Hindawi

Research Article

A Comparative Evaluation of Algorithms in the Implementation of an Ultra-Secure Router-to-Router Key Exchange System

Nishaal J. Parmar and Pramode K. Verma

Telecommunications Engineering Program, School of Electrical and Computer Engineering, University of Oklahoma, Tulsa, OK 74135, USA

Correspondence should be addressed to Nishaal J. Parmar; nishaal.parmar@ou.edu

Received 1 September 2016; Accepted 26 October 2016; Published 12 January 2017

Academic Editor: Vincente Martin

Copyright © 2017 N. J. Parmar and P. K. Verma. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper presents a comparative evaluation of possible encryption algorithms for use in a self-contained, ultra-secure router-to-router communication system, first proposed by El Rifai and Verma. The original proposal utilizes a discrete logarithm-based encryption solution, which will be compared in this paper to RSA, AES, and ECC encryption algorithms. RSA certificates are widely used within the industry but require a trusted key generation and distribution architecture. AES and ECC provide advantages in key length, processing requirements, and storage space, also maintaining an arbitrarily high level of security. This paper modifies each of the four algorithms for use within the self-contained router-to-router environment system and then compares them in terms of features offered, storage space and data transmission needed, encryption/decryption efficiency, and key generation requirements.

1. Introduction

With the rise of globalization, microelectronics, and the information age, the need for rapid, long-distance transmission of unconditionally secure information has never been greater. Whether dealing with military intelligence, corporate secrets shared between two (or more) company offices, remote control of vital national infrastructure components such as power and traffic control systems, or mechanical instructions transmitted to off-site medical devices for telesurgery, device updates, and health reports, there are many situations where the rapid, accurate, and secure transmission of information between two parties is a basic necessity. In extreme cases, alteration or even decryption of this information by unauthorized parties may result in damages of billions of dollars and the lives of others.

Historically, only two encryption schemes have been proposed which offer unconditional security, both unsuitable for practical telecommunications. The first, the one-time pad, proposed by Gilbert Vernam in 1919 [1], utilizes a single-use encryption key equal to the message length which both the sending and receiving parties may use to encrypt and decrypt the message. The disadvantages of this system in a long-term high data rate communication system are obvious, with each message requiring a preshared key equal to the message length. The second, recently proposed unconditional cryptographic system is quantum cryptography, where security is achieved through the laws of quantum mechanics, which allow for very accurate determination of eavesdroppers along a quantum channel, as well as the simultaneous determination of small shared and secure random values. Currently available quantum encryption protocols include BB84, proposed in 1984 by Bennett and Brassard [2], the variant SARG04 [3], and the later-developed B92 [4]. All three solutions, while unconditionally secure, possess severe limitations which make them unsuitable for general commercial use, including reliance on single-photon generators (greatly limiting practical data rate) and, most importantly, the presence of a physical, well characterized quantum channel between endpoints, with a maximum practical distance of a few hundred km. While some research has been proposed in the use of multiphoton quantum sources [5] and channel extension [6], this technology remains extremely expensive and unfeasible for general commercial use.

While unconditional security may be an unachievable goal, it may be realized to an arbitrarily high level via existing symmetric and asymmetric encryption systems. Currently, the most widely used form of global network communication

between two distant parties relies on public key, asymmetric key cryptography such as RSA for transferring symmetric keys. Symmetric encryption systems then use these keys to encrypt the information being transferred. Noteworthy corporations offering SSL certificates with Elliptic Curve Cryptography (ECC), RSA, and DSA support include Symantec (formerly Verisign), GoDaddy, and Comodo.

Although presenting a viable and widely used solution to secure communication, allowing for message encryption and authentication, the security certificate system requires the presence of a trusted third party for the verification of the identity and legitimacy of certificate owners. The compromise of or loss of trust in such a third party, or the inability to contact the distribution network at need, may result in a large-scale breakdown of reliable and secure communications [7]. Furthermore, the increasingly large RSA key length requirements of public certificates to guarantee secure communication may be a barrier to practical implementation on limited-resource devices.

A novel proposal [9] for a secure communication system not requiring a third party for key generation and distribution involves instead a system of paired routers, communicating securely on any standard classical channel. Each router pair would be factory-initialized with a shared secret, enabling direct secure communication between the two regardless of network distance or security, with shared transmitted data on each end decrypted and used as the basis of further secure key generation as necessary. While the original system uses a variant of the discrete logarithm problem to introduce the nonlinearity necessary in a secure encryption system, nonlinearity introduced via other encryption methods offers alternative advantages. This paper first examines the originally proposed discrete logarithm-based encryption system and then proposes and compares other more commonly used encryption systems which may be used in this entirely self-contained environment, including RSA, ECC, and AES based encryption.

2. Related Work: Discrete Logarithm

The encryption system initially proposed in [9] is a variant of the discrete logarithm problem. This problem states that for the equation $b = a^i \bmod(p)$ if a user knows a, I, and p, computing b is computationally trivial. If, however, only b and a are known (and p), then there is no efficient algorithm to compute i.

Under the proposed encryption system, the sender, Alice, and the receiver, Bob, choose a large prime p and its primitive root a. These values may be public. Additionally, Alice secretly chooses two random positive numbers $x, i_1 < p$, Bob secretly chooses one random positive number $i_2 < p$, and both Alice and Bob know a shared secret random number R.

(1) Alice calculates $K = a^x \bmod(p)$ and $L_1 = a^{Ri_1} \bmod(p)$ and sends L_1 to Bob.

(2) Bob, knowing the value of R but not i_1, calculates $L_2 = a^{R(i_1+i_2)} \bmod(p)$ and sends L_2 to Alice.

(3) Alice, who knows the value of i_1 and R but not i_2, calculates $L_3 = a^{R(x+i_2)} \bmod(p)$ and sends L_3 to Bob.

(4) Bob, who knows the value of i_2, may easily recover x from L_3 and use it to calculate Alice's chosen key $K = a^x \bmod(p)$, which is used to encrypt further communications.

As these 3 transmitted equations involve a total of 4 unknowns to any intercepting party (R, x, i_1, i_2), determining the key for anyone who is not Alice or Bob is a nontrivial task, equivalent to finding the exponential of a discrete log problem.

Although initialization of the system requires shared public values of p and a and a shared secret random value R, once a key has been used for a certain length of time, decrypted data transmitted between Alice and Bob may be used by both parties in an algorithm to determine a new R value, and Alice or Bob may propose a new key exchange. With each iteration in a cycle, new values of x, i_1, and i_2 are arbitrarily chosen, while R is partially updated with a shared algorithm applied to the decrypted data transmissions of the previous iteration. With each complete cycle, the R value chosen will be completely replaced from the same iteration of the previous cycle. Further details may be found in [9].

If the key transfer protocol is not completed successfully, whether due to data loss or due to malicious interference, it may be necessary to reinitialize the system via use of another preshared secret R.

Storage requirements for this system involve a preshared secret of length R. Although no minimum length is required for R, for increased security, it should be assumed that R is relatively large, at a minimum approaching the approximate length of p. p itself should be a large prime, in order to deter brute force attacks. Processing time for this encryption system for both encryption and decryption is relatively trivial, involving multiple multiplication, exponentiation, and mod operations. As both endpoints share a common key K, this system does not allow for external message authentication or differentiation between messages originating from Alice or Bob.

The most efficient attack currently used on the general case of the discrete logarithm problem is the number field sieve [10], arriving at a solution for a prime number n in $L_p[1/3; 3^{2/3}]$ (this is approximately $e^{(2.08+O(1))(\log n)^{1/3}(\log\log n)^{2/3}}$). The security provided may thus be directly compared to that of RSA, which also may be most efficiently defeated via the general number field sieve, although discrete logarithms offer slightly more protection for a given key size. A quantum system, once it exists, may use Shor's algorithm to solve this problem in polynomial time [11].

3. Alternative I: RSA

The RSA algorithm has the advantage of being one of the most widely used and studied encryption methods today and is extremely elegant, simple, and well-tested. As the default algorithm used by many SSL providers, as well as the basic

public key encryption scheme most others are compared to, RSA is used here as a baseline for the comparison of other encryption methods, even though it is not as storage-efficient or processing-efficient as other algorithms studied and requires the use of longer key lengths for equivalent security. Current commonly used RSA key lengths include 1024 and 2048 bits.

The basic principle of RSA security rests on the theory that it is extremely difficult to factor the product of two large prime numbers into its constituent factors. Each individual in the RSA network must create 2 complimentary keys, commonly referred to as a public key and a private key, with each key able to decrypt messages enciphered using its compliment. To create this key pair, Alice and Bob must each do the following [1]:

(1) Choose two similar large prime numbers p and q, which are within a few digits of each other in length. p and q are multiplied together to form a modulus n.

(2) An integer e is chosen such that $e < (\varphi(n))$ and e and $\varphi(n)$ are coprime. ($\varphi(n) = n - p - q + 1$). A common value of e is 65,537 ($2^{16} + 1$). The public key consists of n and e (modulus and public key exponent).

(3) The modular multiplicative inverse of $e \bmod(\varphi(n))$, $d \equiv e^{-1}(\bmod(\varphi(n)))$ is calculated, and the private key consists of n and d (modulus and private key exponent).

(4) Message encryption may then be expressed, using the one key, as $C = (m^{K1})(\bmod n)$, while decryption uses the other key as $p = C^{K2}(\bmod n)$.

Typically, as the sending party must know the recipient's public key, as well as their own private key, RSA is not used within a self-contained system. Key generation for large primes may also be time consuming and resource intensive. Instead, third-party organizations must exist and are trusted to verify that a given public key corresponds to the stated owner's private key. Issued certificates linking a public key and verification of its owner's identity are generally valid for a set length of time, after which a new key must be generated and a new certificate request verifying the key's owner must be submitted to the central verification authority.

As our proposed router system must be self-contained after initial manufacture, this third-party verification method is not feasible, and we cannot rely on external communication for the identity verification of new public key data, requiring a slight modification of the standard RSA system. Instead, Alice's router will need to be initialized with prestored values for Alice's private key and Bob's public key, and Bob's will have Alice's public key and his own private key. In this scenario, it is not necessary for either party to know their own public key, and all 4 keys are kept private within the network.

Encryption and decryption function as standard RSA operations, with Alice encrypting data with Bob's public key and Bob decrypting data with his private key, and vice versa. After a data threshold is exceeded, Alice and Bob will both calculate new RSA key pairs and encrypt and send their new public keys using the old keys, with this encryption further acting as identity verification previously requiring a third party. For example, Alice's new public key would be encrypted first with her old private key for authentication and identity verification and then with Bob's old public key for security; then it will be sent to Bob. Bob would decrypt data using his own old private key and then Alice's old public key. Once both parties have received the new keys, all data will be transmitted using these. This system would allow for the use of RSA indefinitely, with rapid key updates, without the necessity of a third party. In the event of a communication failure due to data loss or malicious action, it may be necessary to switch to a new preshared certificate pair and begin the process again.

Storage requirements for an n-bit RSA system are comparatively large, as larger key lengths are needed to assure equivalent security. Specifically, each router using this n-bit RSA algorithm will need to store 1 public and 1 private key, each consisting of an n-bit modulus and a smaller exponent (also of maximum length about n) for maximum total requirement of $4n$ bits per router. Processing time for RSA is also comparatively long, due to the larger key lengths and exponentiation operations required. The security of RSA is based upon the difficulty of the factorization problem. As with the discrete logarithm attack, the current approach to integer factorization involves the general number field sieve algorithm [12], which for an integer n will arrive at a solution in $L_p[1/3; 1.923]$, that is, $(e^{(1.923+O(1))(\log n)^{1/3}(\log\log n)^{2/3}})$. A quantum computer, should it ever exist, may factor large integers in polynomial time [11].

Although it is obvious that RSA offers several disadvantages when compared to other symmetric and asymmetric ciphers, it also offers at least one key advantage when compared to the other algorithms herein: message authentication. Unlike discrete logarithm, ECC, or AES encryption, since neither Alice nor Bob knows the other individual's private key, it would be possible for a third-party external audit, given hardware access to both router keys and all traffic sent, to determine the sender of all encrypted data. Using the other encryption systems, given the encrypted data alone, it is possible to determine that either Alice or Bob sent a message, but not to authenticate which one encrypted the data.

4. Alternative II: AES

AES, based upon the Rijndael cipher, was announced by the National Institute of Standards and Technology in 2001 and was shortly thereafter approved as an accepted encryption standard by the United States Federal Government. AES, similar to its predecessor, DES, is a symmetric block cipher, using a shared secret key to encrypt a data stream one block at a time. In AES, each 128-bit data block undergoes 10–14 rounds (depending on key length) of permutations, substitutions, and additions [1]. AES is an extensively used and studied algorithm and like most symmetric ciphers offers advantages in terms of required processing power, processing time, and key length when compared to asymmetric ciphers such as RSA and ECC. The simplicity of each round enables simple and rapid implementation on any 8-bit processor, while the chaining of multiple rounds per block provides excellent

security. The AES algorithm itself is quite straightforward to implement within hardware, and hardware AES optimization is currently already present in many modern, commercially available processors, including current processors from Intel, AMD, and Qualcomm, making this an excellent algorithm choice for use with existing components.

To modify AES for use in our closed system, Alice and Bob's routers will both require a single preshared AES key and a reliable PRNG. Initial communication will be made using the preshared key. After a data threshold has been reached, similar to the discrete logarithm system, Alice and Bob will input the decrypted data into an algorithm (such as a cryptographic hash function) to generate a random value R. This value will be used as a PRNG seed on both systems to generate identical intermediate keys of the desired AES key length. To compensate for any bias in the data used to generate R (similar data between data cycles may lead to a smaller PRNG seed pool), the intermediate key may be XOR'd with the previous AES key to generate a new, random shared secret key by which further communication will be encrypted.

As mentioned earlier, AES offers efficient processing time, and the storage requirements for this system are minimal, requiring a single preshared key to be saved on each of the two end routers, much shorter than a security-equivalent RSA key pair. No effective cryptanalytic attacks are currently known against AES, with the current best attacks only a few orders of magnitude above the worst-case brute force scenario and requiring infeasibly large amounts of storage space [13]. Unlike asymmetric encryption algorithms, AES is likely resistant to attacks by theoretical future quantum computers. In the event of a communication failure due to data loss or malicious action, it may be necessary to switch to a new preshared key and begin the process again.

5. Alternative III: ECC

Elliptic Curve Cryptography (ECC) is an asymmetric cryptographic system, which uses a variant of the discrete logarithm problem as applied to points in an elliptic curve group as the core of its security. Many consumers have recently begun adopting ECC as an alternative to RSA, due to its efficiency in both key size and processing requirements. Careful choice of the ECC curve is necessary to avoid potential security hazards.

In Elliptic Curve Cryptography, first a curve is chosen, with variables and coefficients restricted over either the finite field GF(2^m) of the form $y^2+xy = x^3+ax^2+b$ or a prime curve over Z_p and modulo p where variables and coefficients range from 0 to $(p-1)$ of the form $y^2 \bmod p = (x^3 + ax + b) \bmod p$.

In the prime curve case, there are a limited number of nonnegative integer points between $(0, 0)$ and $(p-1, p-1)$ which satisfy any given elliptic curve values for a and b. Similarly, for the finite field case, there will be a limited number of (x, y) integer values that lie on the curve for any given values of a and b.

These points are used to define a finite abelian group, with rules for addition defined specifically for the abelian group, similar to modular multiplication in conventional algorithms. Likewise, multiple additions are preformed similarly to modular exponentiation. Using abelian group rules, given two points M and N, $M = kN$ is easily calculated given k and N but difficult to calculate given M and N, forming the one-way trapdoor function at the basis of elliptic cryptography.

Generally, the curve parameter values of a, b and z, C and n are made public and often correspond to one of several well-studied elliptic curves. a and b are the coefficients discussed earlier, forming the curve $E_z(a, b)$, where z is an integer in the finite field 2^m (finite field curve) or a large prime number (prime curve). A base point C is picked such that the smallest positive integer n that satisfies $nC = 0$ is very large. With all curve parameters defined, Alice and Bob may begin the key selection process [1].

(1) Alice and Bob both choose secret integers I_A and I_B less than n as their private keys.

(2) Public keys are generated according to $p_A = I_A \times C$ and $p_B = I_B \times C$ and shared with each other.

(3) A common secret key is generated my multiplying the known private key with the opposite public key, with $I_A \times p_B = I_B \times p_A$.

(4) To encrypt or decrypt data, the data is first encoded as a point M on the elliptic curve and then sent as a ciphertext message as a pair of points $(kC, M + kp)$ with k as any chosen positive integer and decrypted with the matching private key using $(M_x I - M_y)$.

Modifying this system to function in our self-contained router environment involves a process similar to that used for RSA. All curve parameters are assumed to be publicly known, and use of a known secure curve is assumed. Each router must be initialized with secret data corresponding to its own private key and the public key of the other router. Again, it is not strictly necessary for each party to know or retain its own public key, and, in any case, all 4 key values are kept secret within the network.

Encryption and decryption function as standard ECC operations, with Alice encrypting data with Bob's public key and Bob decrypting data with his private key, and vice versa. After a data threshold is exceeded, Alice and Bob will both calculate new public and private ECC keys, choosing new secret integers, and encrypt and send each other their new public keys using their old private keys. Once both parties have received the new keys, all data will be transmitted using these. This system would allow for the use of ECC indefinitely, with rapid key updates, without the necessity of a third party. In the event of a communication failure due to data loss or malicious action, it may be necessary to switch to a new preshared certificate pair and begin the process again. Unlike in RSA, the use of a common secret key prevents message authentication via external audit.

Storage requirements for ECC involve two large integers of size n or smaller, corresponding to the public and private keys, on each router, for a total maximum storage capacity of $2n$ per shared secret per router. Key lengths used are much shorter than those needed for equivalent RSA or

discrete logarithm security levels, about double the size of that found in symmetric encryption systems. Likewise, while not quite as processing-efficient as a symmetric cryptosystem, ECC offers large performance gains when compared to RSA. The best known attack to ECC is Pollard's Rho [14] which may be paralyzed and needs relatively little memory but is nevertheless not computationally feasible for currently used curve parameters. As with other public key protocols, ECC is expected to be vulnerable to attack by quantum computers, once such exist.

6. Algorithm Comparison

The RSA, ECC, AES, and discrete logarithm protocols may each provide an arbitrary level of security, determined by the length of the encryption keys used for each algorithm [8]. Figure 1 visually illustrates the required key length needed by various encryption algorithms in order to achieve a level of security comparable to a specified RSA key length (e.g., to achieve the same level of security provided by 2048-bit RSA encryption, AES requires only a 112-bit key). In the case of the discrete logarithm method, the equivalent key length of the prime p used was determined using the general number field algorithm as compared to RSA key lengths and was found to be approximately equal in requirement with RSA key N equivalent to a discrete log key $0.84N$ (less than one-bit difference). ECC and AES hold clear advantages here over RSA and discrete log methods, as key sizes for the latter two increase rapidly as increased security is needed, while the key length : security ratio remains relatively linear for ECC and AES. The longer key lengths of RSA and discrete log will also require additional bandwidth for public key transfer, compared to shorter ECC public keys, and no additional bandwidth overhead is required for AES.

Storage requirements for preshared secret data per router (ignoring overhead and indexing values), as outlined by the modified algorithms described earlier, are as follows:

(1) n-bit RSA requires a maximum of $4n$ bits per secret.

(2) n-bit ECC requires a total of $2n$ bits per secret.

(3) n-bit AES requires a single stored n-bit key.

(4) n-bit discrete log method involves a preshared secret R, assumed to be of maximum length n.

Using these values, in combination with the key length requirements illustrated in Figure 1, it is possible to calculate the minimum storage requirements of each router for preshared secret data. For example, from Figure 1, we see that a 2048-bit RSA or discrete logarithm key is the equivalent of a 224-bit ECC key, or a 112-bit AES key. Each shared secret stored by the router at this security level would thus require a maximum of $(2048 * 4) = 8192$ bits for RSA and 2048 bits for discrete log but only $(224 * 2) = 448$ bits for ECC, or 112 bits for AES. Using these calculations, Figure 2 illustrates the total number of preshared secrets which may be stored per gigabyte of memory for any given security level and encryption algorithm (e.g., 8,000,000,000/8192 = 976,562 shared secrets per GB for 2048-bit RSA, or over 70 million shared secrets per GB for the equivalent 112-bit AES).

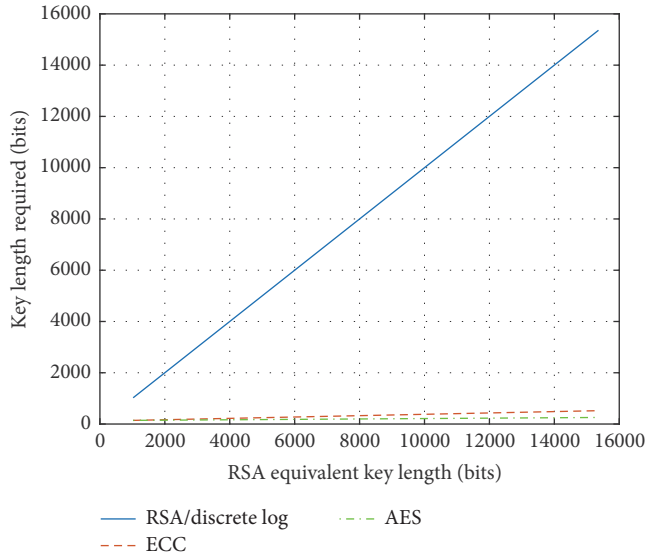


Figure 1: Key length versus security for AES, ECC, RSA, and discrete log. Data source: National Security Agency, Central Security Service [8].

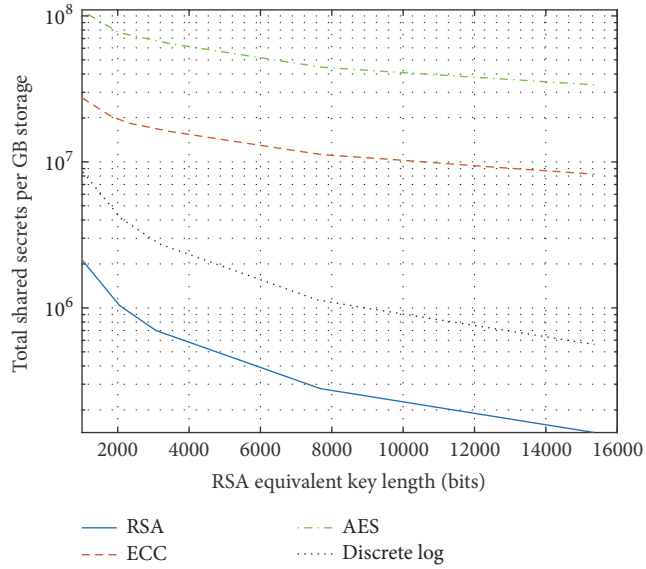


Figure 2: Router preshared secret storage requirements.

As calculated in Figure 2, a single GB of router secret data storage allocation may hold hundreds of thousands of shared secrets even when using the inefficient RSA algorithm at the 7680-bit security level. When using AES, millions of shared secrets may be stored in this space. Regardless of algorithm choice, shared secret storage space is unlikely to be a limiting factor in practical router implementation.

Encryption and decryption performance for the various algorithms are difficult to measure and are heavily influenced by system architecture and software/hardware optimizations. Generally, however, symmetric key ciphers such as AES will offer the fastest encryption and decryption times. ECC offers dramatically superior key pair generation performance

compared to RSA, with the large primes generated for RSA requiring several orders of magnitude more time when compared to a much smaller ECC key, especially at RSA bit lengths of 2048 and above. In router systems with frequent key refreshes this could be a potential issue. Additionally, manufacturing hardware may struggle to fill even a modestly sized storage chip with unique preshared RSA keys (even a 1 GB sized chip may be able to hold hundreds of thousands of preshared RSA certificates!), while even millions of shared symmetric encryption keys would simply involve filling the same chip pair with identical random data. RSA encryption is generally slightly faster than ECC, while ECC decryption may be several times faster than RSA, although both are generally efficient enough not to provide a practical system bottleneck [15, 16]. The discrete log method is assumed to offer a similar processing time as RSA due to similarities in algorithm implementation but will likely take longer due to the multiple exchanges involved.

7. Practical Implementations

The primary limitation on this router-to-router encryption system is the necessity for each router to be factory-manufactured containing shared secret information, enabling secure communication only with its matched counterpart.

This limitation may be partially mitigated by offering routers containing several small storage chip expansion slots. These storage chips would be manufactured in pairs, with each pair stamped with a matching serial number and containing a number of matching shared secret keys. Although each chip should be clearly labeled with its identical match, the actual matching data therein should not be retained after generation by the manufacturer, preventing compromise of manufacturer records from affecting system security.

A single router could thus be configured to securely communicate with a number of endpoints, with each endpoint sharing a unique inserted security chip pair, easily installable and replaceable as needed. Given the low cost of solid-state storage, under any proposed encryption scheme, the number of shared initial secret keys on a single chip would well exceed the lifetime of the router itself, even in a scenario where high data loss over a connection prevents the easy determination of additional keys before another shared hardware key is needed.

8. Conclusion and Future Work

Ultimately, algorithm choice will likely be determined by system needs and the availability of supporting hardware. Whatever algorithm is chosen, it will be necessary to provide preshared secret data to factory-paired communication devices, either built directly into each router pair or provided as paired insertable expansion chips with pregenerated shared encryption keys. Once the initial key is shared, a combination of PRNG values, prior secret data, and decrypted current communication may be used to generate new secure keys on demand, ensuring a regular refresh of the currently used key. While advances in modern solid-state storage make it unlikely that shared secret storage space is ever a practical limitation of the proposed router-to-router key exchange system, algorithm processing efficiency, data efficiency, and key generation time may have a much larger impact on system design.

While discrete logarithm, RSA, ECC, and AES may each be used to provide the necessary nonlinearity for the establishment of a self-contained secure communication channel between two paired hardware devices, RSA and AES offer the most features and most efficient functionality, respectively. If authentication is needed, RSA, the weakest algorithm in terms of key generation and processing efficiency, is the clear choice. The use of RSA will, however, require a great deal of additional key generation time on the router manufacturing end. If, however, authentication is not needed, then symmetric key systems such as the AES exchange proposed offer the most efficient alternative and the only choice which offers more resistance to quantum computing attacks. AES hardware optimization is both extremely efficient and widely available in many currently used commercial processors, resulting in superior encryption, decryption, and processing times. AES key pair data, consisting effectively of a random bitstream, may be much more rapidly generated and preloaded onto devices than RSA, ECC, or discrete logarithm key pairs and provide greater security than equivalent-length asymmetric ciphers. Alternatively, a hybrid of both systems may be used, offering on-demand authentication when needed and efficient nonauthenticated secure communication otherwise.

As a final consideration, as with any digital security, any encryption system is vulnerable to physical hardware compromise. If an attacker is able to gain access to the shared secret data stored on the router's security hardware, even the most secure encryption framework will be compromised, and care must be taken during hardware manufacture and distribution to ensure that these keys are not copied or prematurely accessed.

Competing Interests

The authors declare that they have no competing interests.

References

[1] W. Stallings, *Cryptography and Network Security: Principles and Practice*, Prentice Hall, 6th Edition, 2014, 2011, 2006, pp. 47-48, pp. 264–278, pp. 130–173, pp. 303-304.

[2] C. H. Bennet and G. Brassard, "Quantum cryptography: public key distribution and coin tossing," in *Proceedings of the IEEE International Conference on Computer System and Signal Processing*, Bangalore, India, December 1984.

[3] V. Scarani, A. Acín, G. Ribordy, and N. Gisin, "Quantum cryptography protocols robust against photon number splitting attacks for weak laser pulse implementations," *Physical Review Letters*, vol. 92, no. 5, Article ID 057901, 4 pages, 2004.

[4] C. H. Bennett, "Quantum cryptography using any two non-orthogonal states," *Physical Review Letters*, vol. 68, no. 21, pp. 3121–3124, 1992.

[5] S. Mandal, G. Macdonald, M. El Rifai et al., "Implementation of secure quantum protocol using multiple photons for communication," https://arxiv.org/abs/1208.6198.

[6] Z. Zhao, T. Yang, Y.-A. Chen, A.-N. Zhang, and J.-W. Pan, "Experimental realization of entanglement concentration and a quantum repeater," *Physical Review Letters*, vol. 90, no. 20, Article ID 207901, 2003.

[7] J. Menn, "Key internet operator VeriSign hit by hackers," *Reuters*, 2012.

[8] "The Case for Elliptic Curve Cryptography," National Security Agency—Central Security Service, Jan 2009, http://web.archive.org/web/20150627183730/https://www.nsa.gov/business/programs/elliptic_curve.shtml.

[9] P. K. Verma and M. El Rifai, "An Ultra-secure Router-to-router Spontaneous Key Exchange System," *International Journal of Computer Network and Information Security*, vol. 7, no. 7, pp. 1–9, 2015.

[10] D. M. Gordon, "Discrete logarithms in GF(P) using the number field sieve," *SIAM Journal on Discrete Mathematics*, vol. 6, no. 1, pp. 124–138, 1993.

[11] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Review*, vol. 41, no. 2, pp. 303–332, 1999.

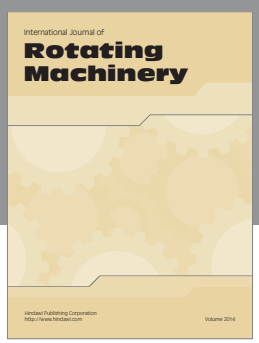
[12] D. Coppersmith, "Modifications to the number field sieve," *Journal of Cryptology*, vol. 6, no. 3, pp. 169–180, 1993.

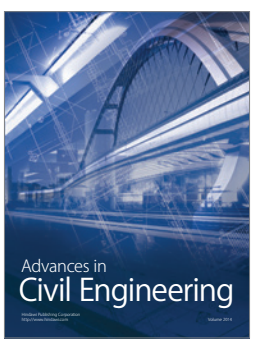
[13] A. Bogdanov, D. Khovratovich, and C. Rechberger, "Biclique cryptanalysis of the full AES," in *Advances in Cryptology—ASIACRYPT 2011: 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4–8, 2011. Proceedings*, vol. 7073 of *Lecture Notes in Computer Science*, pp. 344–371, Springer, Berlin, Germany, 2011.

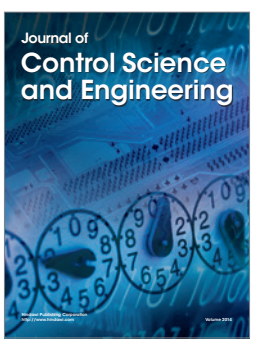
[14] J. W. Bos, M. E. Kaihara, T. Kleinjung, A. K. Lenstra, and P. L. Montgomery, "On the security of 1024-bit RSA and 160-bit elliptic curve cryptography," Tech. Rep. EPFL-REPORT-164549, 2009.

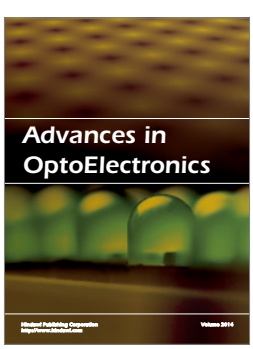
[15] M. Savari, M. Montazerolzohour, and Y. E. Thiam, "Comparison of ECC and RSA algorithm in multipurpose smart card application," in *Proceedings of the International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec '12)*, IEEE, 2012.

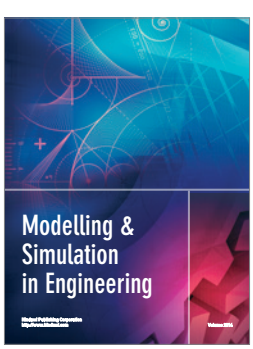
[16] K. Maletsky, "RSA vs ECC comparison for embedded systems," Atmel, 2015, http://www.atmel.com/images/atmel-8951-cryptoauth-rsa-ecc-comparison-embedded-systems-whitepaper.pdf.

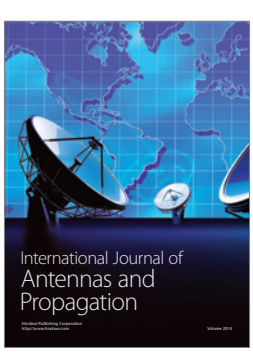


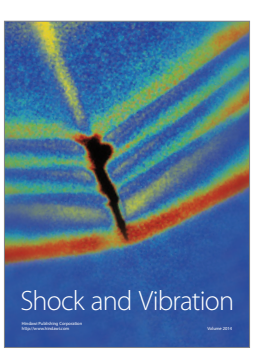


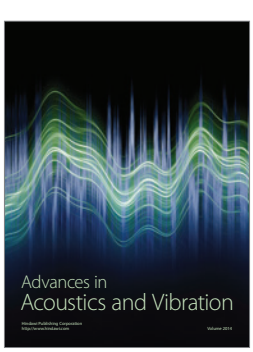












Hindawi

Submit your manuscripts at
https://www.hindawi.com