BAB II
Tinjauan Pustaka

2.1. Faktor – faktor Otentikasi

Metode untuk mengotentikasi manusia berbeda dengan metode untuk mengotentikasi mesin dan program. Hal ini disebabkan karena perbedaan terbesarnya adalah kemampuan manusisa dengan komputer. Komputer sangat ahli dalam melakukan kalkulasi yang cepat dan tepat dan komputer juga memiliki *memory* yang dapat disimpan dan diambil kembali dalam ukuran informasi yang besar. Di lain sisi, manusia tidak memiliki kemampuan seperti itu sehingga diperlukan metode yang berbeda untuk mengotentikasi manusia [3].

Seluruh pendekatan otentikasi manusia menggunakan minimal satu dari tiga faktor otentikasi berikut:

Something you know (contoh: kata sandi). Ini merupakan hal yang paling standar dalam otentikasi yang digunakan untuk manusia. Kita mengugnakan kata sandi hampir setiap hari untuk mengakses sistem. Sayangnya, untuk manusia faktor *something you know* dapat dengan mudah menjadi sesuatu yang dilupakan, dan jika faktor ini dituliskan, orang dapat mengetahuinya [3].

Something you have (contoh: *smart card,* kunci mobil). Bentuk otentikasi ini menghilangkan masalah manusia akan melupakan faktor SYK. Namun, sebagai gantinya, sebuah objek untuk khusus harus dibawa ketika ingin melakukan otentikasi. Masalah yang dapat timbul dari bentuk ini adalah objek tersebut memiliki potensi untuk dicuri dan dapat dapat digunakan untuk mengotentikasi sistem yang sama [3].

Something you are (contoh: sidik jari). Otentikasi yang menggunakan komponen intrinsik yang melekat di manusia. Dengan menggunakan anggota tubuh manusia tidak perlu membawa objek khusus dan tentu lebih sulit kehilangan anggota tubuh dari pada objek yang perlu di bawa-bawa seperti kunci dan dompet. Akan tetapi, sensor biometrik hingga saat ini masih belum tergolong murah dan sering kali tidak akurat [3].

2.1.1. Something you know (SYK)

Something you know berarti mengetahui sesuatu dan sesuatu itu biasanya berupa kata sandi. Sehingga sebuah rahasia yang hanya diketahui

oleh sistem dan manusia itu membedakan satu individu dengan yang lainnya. Beberapa teknologi dari metodologi ini adalah:

1. PIN (*Personal Identification Number*)

   PIN biasanya berupa kata sandi dengan panjang 4 s.d. 6 digit angka. Di-*input* ke dalam sistem untuk memeroleh akses dari sistem tersebut. Dewasa ini banyak digunakan pada sistem perbankan seperti ATM, EDC, *internet banking*, dan *mobile banking*.

2. Kata sandi

   Kata sandi merupakan pilihan yang paling banyak digunakan untuk mengotentikasi pengguna walaupun pada faktanya peretasan kata sandi termasuk cara yang paling mudah bagi peretas untuk mengakses sistem. Kata sandi sangat bergantung pada kerahasiaan. Peretas memiliki banyak cara untuk mendapatkan kata sandi seseorang. Peretas dapat mencuri kata sandi dengan mudah dari dengan hanya mengintip ketika diketikkan, penggunaan *password-cracking software*, hingga mencuri kata sandi melalui jaringan dengan penggunaan perangkat lunak *remote cracking utilities* atau *network analyzers*[4] [5]. Secara umum terdapat dua klasifikasi dari celah keamanan kata sandi:

   a. Celah keamanan pengguna

      Kelemahan ini termasuk akan kurangnya kesadaran akan pentingnya kata sandi yang baik oleh pengguna atau organisasi. Sudah menjadi hakikat manusia untuk serba praktis, orang-orang cenderung membuat kata sandi yang mudah digunakan dan diingat seperti “password”, tanggal lahirnya atau kerabat dekatnya, nama orang, dan nama hewan peliharaan. Selama pengguna tidak pernah diedukasi mengenai kata sandi yang baik, kata sandi mereka pada umumnya lemah dan mudah ditebak, jarang diganti, dan digunakan ulang pada banyak sistem, ditulis dan disimpan pada tempat yang tidak aman. Hal ini mengakibatkan kata sandinya mudah didapatkan oleh peretas dan jika didapatkan, peretas dapat masuk ke dalam banyak sistem miliknya karena kata sandinya digunakan secara berulang [4] [5].

   b. Celah keamanan teknis

      Celah keamanan ini termasuk metode enkripsi yang lemah, metode penyimpanan kata sandi yang tidak aman. Peretas dapat melakukan *crack*

pada sistem penyimpanan kata sandi yang lemah menggunakan metode *cracking* tertentu. Penggunaan kata sandi yang lemah memudahkan peretas dapat mendapatkan kata sandi. Banyak sekali program *password cracking* yang memanfaatkan celah keamanan dari lemahnya kata sandi. Program-program ini dapat melakukan pekerjaannya dan mendapatkan kata sandi apa pun tergantung pada waktu dan kecepatan pemrosesan. Selain itu, perangkat lunak yang menyimpan kata sandi pada memori dan basis data yang mudah diakses juga mudah dieksploitasi bagi peretas (penyimpanan kata sandi pada browser). Beberapa celah keamanan pada kata sandi contohnya *social engineering, shoulder surfing, inference, weak authentication, password cracking software, keystroke logging, network analyzer, phishing*, dan *man in the middle attack*.

2.1.2. Something you have (SYH)

"*Something you have"* memverifikasi bahwa pengguna memiliki suatu benda fisik untuk mengakses sistem yang diinginkan. Berikut adalah beberapa protokol yang sering digunakan:

1. SMS

   Prinsip dasar otentikasi SMS sama dengan OTP tetapi pengguna harus mendaftarkan nomor telepon pengguna dan *online services* akan mengirim SMS yang berisi OTP. Otentikasi SMS memiliki kekurangan yang sama dengan OTP dan ditambah beberapa hal berikut: SMS memerlukan biaya yang relatif mahal per SMS; memerlukan jangkauan jaringan telepon; isu privasi karena *online services* harus mengetahui nomor telepon pengguna; SIM dapat dilakukan *reissue* dengan mudah; protokol SMS tidak aman dan dapat disadap; proses pengiriman SMS tidak terkirim secara langsung (menimbulkan delay); otentikasi SMS tidak punya standar; dan otentikasi SMS sulit digunakan.

2. One Time Passwords (OTP)

   OTP menggunakan suatu perangkat untuk membuat OTP. Perangkat tersebut dapat berupa aplikasi di *smart phone* maupun alat seperti token. Pengguna harus lebih dulu mendaftarkan aplikasi atau perangkat OTP ke server. Ketika ingin melakukan log in, pengguna harus menggenerasi OTP (dapat berupa HOTP atau TOTP) dan kemudian dikirimkan ke server.

Walaupun bekerja dengan cukup baik hingga saat ini, OTP telah terbukti memiliki celah keamanan. Penggunaan OTP masih dapat dilakukan serangan *phishing* dan serangan MiTM. OTP juga memerlukan proses memasukkan *shared key* ke sistem dan waktu sinkronisasi yang relatif singkat sehingga membuat OTP sulit digunakan bagi orang awam. Pengguna juga harus memiliki aplikasi tertentu atau perangkat khusus untuk menggenerasi OTP, dan hanya bekerja secara *one per site*.

2.1.3. Something you are (SYA)

"*Something you are*" memverifikasi bahwa pengguna adalah benar pengguna yang dimaksud dengan merepresentasikan fitur biometrik seperti iris, sidik jari, wajah, geometri tangan, suara, dan gerak gerik tubuh.

2.2. Multi Factor Authentication

Multi factor authentication (MFA) adalah metode kontrol akses sehingga pengguna hanya dapat diberikan akses setelah terbukti dapat menyediakan beberapa bukti ke mekanisme otentikasi. Biasanya berupa dua dari tiga kategori berikut: informasi (*something they know)*, kepemilikan (*something they have)*, dan sesuatu yang melekat pada dirinya *(something they are)* [6].

Penggunaan MFA untuk membuktikan keaslian identitas pengguna didasarkan pada besarnya kemungkinan bahwa pihak yang tidak memiliki otorisasi sulit untuk menyediakan faktor-faktor tersebut. Contoh faktor-faktor tersebut adalah:

1. Objek fisik yang dimiliki pengguna; seperti USB stick dengan token rahasia, kartu bank, kunci, dll.
2. Rahasia yang diketahui pengguna; seperti kata sandi, PIN, TAN, dll.
3. Karakter fisik dan mental pengguna; seperti sidik jari, iris, suara, kecepatan mengetik, dll.

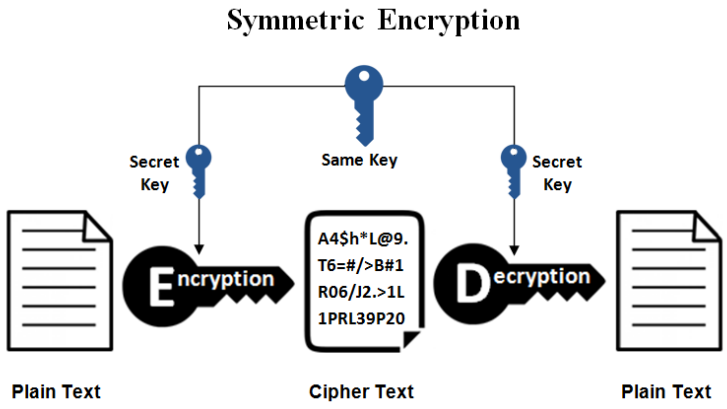
2.3. Enkripsi

Dalam kriptografi, enkripsi adalah proses menyandikan pesan atau informasi sedemikian rupa sehingga hanya pihak yang berwenang yang dapat membaca pesan atau informasi tersebut. Pada enkripsi digital, informasi yang ingin diamankan biasanya disebut *plaintext*, dan dienkripsi menggunakan algoritma enkripsi yang dinamakan *cipher* yang menghasilkan *cipher text* yang hanya bisa dibaca jika didekripsi. Algoritma enkripsi biasanya menggunakan *pseudo random encryption key* yang digenerasi oleh sebuah algoritma. Secara prinsip *cipher text*

memang dapat dilakukan dekripsi tanpa menggunakan *key*. Namun, algoritma enkripsi yang baik akan membutuhkan sumber daya dan kemampuan yang tinggi untuk diretas sedangkan untuk pemilik *key* yang berwenang akan dengan mudahnya mendekripsi *cipher text* untuk mendapatkan informasi yang diinginkan [7].

2.3.1. Symmetric Key Encryption

Pada *symmetric cryptography*, *key* yang sama digunakan untuk proses enkripsi maupun dekripsi. Keuntungan penggunaan *symmetric key encryption* adalah proses pembuatan *symmetric key* yang kuat relatif tidak menghabiskan banyak sumber daya, proses enkripsi dan dekripsi menggunakan *symmetric key* juga relatif tidak berat untuk diproses. Sehingga menggunakan *symmetric key cryptography* lebih ramah penggunan karena waktu delay tidak terlalu berarti. *Symmetric key cryptography* juga menyediakan faktor *authentication* karena jika bukan dengan *symmetric key* yang sama, tidak ada yang dapat mendekripsi suatu *cipher text*. Jadi selama *symmetric key* bersifat rahasia antara dua pihak yang menggunakannya, dapat disimpulkan bahwa komunikasi tersebut aman [8]. Namun, jika dengan cara tertentu suatu pihak mendapatkan *key* tersebut, aspek *confidentiality* dan *authentication* akan terkena dampak. Suatu pihak tak berwenang dengan *symmetric key* yang sama tidak hanya dapat mendekripsi pesan dan melihat isinya tetapi juga dapat mengirim pesan dan berpura-pura menjadi suatu pihak dengan cara mengenkripsi pesan tersebut dan dikirimkan ke pihak yang lain.



Gambar 2.1 *Symmetric key cryptography* [9]

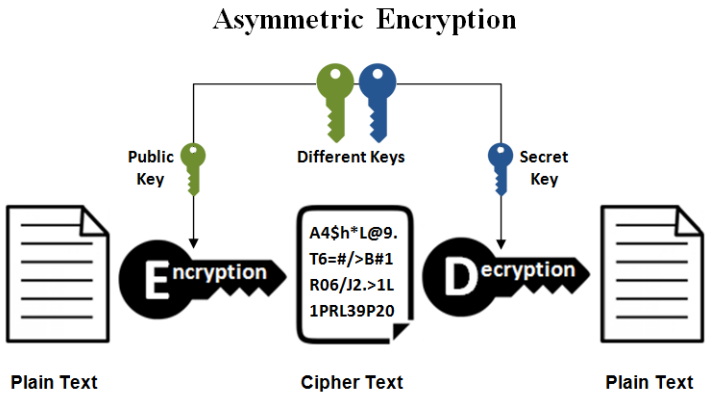
Algoritma *symmetric key encryption* yang pertama kali dikembangkan di Amerika Serikat adalah DES (*Data Encryption Standard)*, disetujui secara luas untuk digunakan sebagai *symmetric key encryption* pada

tahun 1970. DES menggunakan *key* 56 bit. Namun, dikarenakan computer telah menjadi sangat cepat, DES tidak lagi dianggap aman walaupun dapat memiliki lebih dari 70 ribu trilyun kombinasi yang berbeda [8].

DES kemudian digantikan oleh algoritma yang lebih modern dan aman yaitu AES (*Advanced Encryption Standard)* yang menggunakan *key* dengan panjang 128, 192, atau 256 bit. Dipercaya bahwa AES cukup untuk menjadi standar enkripsi *symmetric key* untuk beberapa waktu ke depan. AES dengan 128 bit *key* saja dapat memiliki lebih dari 300,000,000,000,000,000,000,000,000,000,000 kombinasi yang berbeda [8].

2.3.2. Assymetric Key Encryption

Kriptografi asimetrik menggunakan satu pasang *key* yang terdiri dari *public key* yang dapat disebar ke pihak mana saja dan *private key* yang hanya boleh diketahui oleh si pemilik. Dengan satu pasang *key* ini, dua aspek keamanan terpenuhi yaitu *authentication* dan *encryption.* Aspek *authentication* tercapai jika *public key* digunakan untuk memverifikasi pesan yang terenkripsi dengan *private key* sedangkan aspek *encryption* tercapai ketika *private key* dapat mendekripsi pesan yang terenkripsi oleh *public key* [10].



Gambar 2.2 Asymmetric key cryptography [9]

Adapun beberapa algoritma *asymmetric key cryptography* yang terkenal seperti DSA, ECC, RSA, Diffie-Hellmann Key Exchange. Namun, yang paling sering digunakan adalah algoritma RSA (Rivest-Shamir-Adleman) yang digunakan pada protokol seperti SSL/TLS sebagai penyedia

keamanan dalam jaringan komputer. *Key* RSA biasanya memiliki panjang 1024 atau 2048 bit [11].

2.3.3. RSA

RSA pertama kali dikenalkan oleh Ron Rivest, Adi Shamir, dan Leonard Adleman dari MIT pada tahun 1977. *Public key cryptography* atau juga dikenal dengan *asymmetric cryptograp*hy menggunakan dua *key* tetapi terhubung secara matematis, satu *key* dinamakan *public key* dan satu lagi *private key*. *Public key* dapat disebar ke pihak mana pun dan *private key* hanya boleh diketahui pembuat. Pada RSA, *public key* dan *private key* dapat mengenkripsi pesan dan untuk mendekripsi *cipher text*-nya harus digunakan *key* yang berlawanan dari *key* yang melakukan enkripsi. Kemampuan ini adalah alasan RSA menjadi algoritma *asymmetric cryptography* yang paling banyak digunakan. RSA menyediakan aspek keamanan pada bidang komunikasi dan penyimpanan data dalam aspek *confidentiality*, *integrity*, *authenticity*, dan *non-reputability* [12].

Protokol pada layer yang lebih seperti SSH, OpenPGP, S/MIME, dan SSL/TLS juga menggunakan RSA untuk proses enkripsi data dan fungsi *digital signature* karena diperlukannya metode untuk melakukan koneksi yang aman pada jaringan yang tidak aman seperti internet [12].

Ide dasar keamanan pada algoritma RSA terdapat pada kesukaran untuk memfaktorkan dua bilangan bulat yang besar yang merupakan perkalian dari dua buah bilangan prima yang besar. Mengkalikan dua buah bilangan prima itu mudah tetapi untuk menentukan bilangan prima apa yang digunakan untuk mencapai hasil tersebut itu dianggap sangat susah bahkan tidak mungkin karena masalah keterbatasan sumber daya pemrosesan bahkan dengan super komputer yang ada sekarang.

Masalah yang paling kompleks pada RSA adalah proses penggenerasian *public key* dan *private key*. Dua buah bilangan prima yang besar p dan q digenerasi menggunakan algoritma *Rabin-Miller primality test.* Modulus n dikalkulasi dengan mengalikan p dan q. Nomor ini kemudian digunakan pada *public key* dan *private key* dan hal inilah yang membuat adanya koneksi di antara dua buah *key*.

Secara garis besar, ada pun algoritma RSA dapat dijabarkan sebagai berikut [13]:

1. Penggenerasian *public key* dan *private key*

   Proses penggenerasian *public key* dan *private key* mengikuti langkah-langkah algoritma sebagai berikut:

   a. Pilih dua buah bilangan bulat prima yang besar **p** dan **q**
   b. Hitung **n=pq, n** adalah modulus dari *public key* dan *private key*
   c. Hitung totient: $\phi(n) = (p-1)(q-1)$
   d. Pilih integer e di mana $1 < e < \phi(n)$ dan e koprima dengan $\phi(n)$ contoh: e dan$\phi(n)$ tidak memiliki faktor selain 1; gcd $= \big(\mathrm{e}, \phi(n)\big) = 1$. $\boldsymbol{e}$ dirilis ke publik sebagai *public key*.
   e. Hitung $\boldsymbol{d}$ sehingga memenuhi kekongruenan $de \equiv 1 \,(\mathrm{mod}\big(\phi(n)\big)$ contoh: $de = 1 + k\phi(n)$ untuk integer **k**. **d** kemudian disimpan sebagai *private key*.

2. Enkripsi

   Public key yang berupa (n & e) dirilis ke publik. Jika diinginkan pengiriman pesan M maka:

   a. Pesan M harus diubah menjadi m di mana m lebih kecil dari n dengan menggunakan protokol reversibel yang telah disetujui sebelumnya yaitu *padding scheme*.
   b. Setelah itu komputasi *cipher text (c)* sehingga memenuhi c $= m^e \, mod \, n$

3. Dekripsi

   Pesan m dapat dikembalikan dari pesan c (m yang telah dienkripsi) dengan menggunakan d melalui proses sebagai berikut:

   a. Hitung kembali m sehingga memenuhi

      $$\mathrm{m} = c^d \, mod \, n$$

   b. Dengan demikian didapatkan m sehingga dapat diproses lebih lanjut untuk mendapatkan pesan asli M.

      Proses dekripsi dapat berlangsung karena atribut matematik sebagai berikut:

      i. $c^d \equiv (m^e)^d \equiv m^{ed} \; (mod \, n)$

ii. Dan dikarenakan

$$ed \equiv 1\ mod(p-1) \text{ dan } ed \equiv 1\ mod(q-1)$$

iii. Berdasarkan *Fermat's little theorem* maka didapatkan

$$m^{ed} \equiv m\ (mod\ p) \text{ dan } m^{ed} \equiv m\ (mod\ q)$$

iv. Dikarenakan p dan q adalah bilangan prima maka dengan mengaplikasikan *Chinese remainder theorem*, dari dua persamaan tersebut didapatkan

$$m^{ed} \equiv m\ (mod\ pq)$$

v. Sehingga didapatkan $c^d \equiv m\ (mod\ n)$

2.4. Android KeyStore

Android keystore memungkinkan pengguna untuk menyimpan *cryptographic keys* di dalam suatu kontainer khusus dengan tujuan agar lebih sulit untuk mengekstraknya dari perangkat. Ketika *key* sudah di dalam *keystore*, *key* dapat digunakan untuk melakukan proses kriptografi tetapi dengan *key* tidak dapat diekspor keluar dari sistem. Android Keystore provider dikenalkan pada Android 4.3 (API 18) [14].

Sistem Android Keystore memproteksi *key* dari penggunaan yang tak terotorisasi. Pertama, *Extraction Prevention*, Android Keystore mencegah penggunaan tak terotorisasi untuk *key* di luar perangkat Android dengan cara mencegah ekstrasi *key* dari proses aplikasi dan dari perangkat Android secara keseluruhan. Kedua, *Key Use Authorizations*, Android Keystore mencegah penggunaan tak terotorisasi dari *key* pada perangkat Android dengan cara membuat aplikasi menspesifikasikan pengguna yang terotorisasi atas *key* tersebut serta tetap menjalankan restriksi ini di luar proses aplikasi tersebut [14].

2.4.1. Extraction Prevention

Key dari Android Keystore terproteksi dari proses ekstraksi menggunakan dua lapis keamanan [14]:

1. Key tidak pernah memasuki proses aplikasi

   Ketika aplikasi melakukan *cryptographic operations* menggunakan *key* dari Android Keystore, secara *background, plain text, cipher text,* dan pesan yang perlu di verifikasi / enkripsi akan disalurkan ke *system process* yang melaksanakan tugas kriptografinya. Jika *application process* dibajak,

maka penyerang hanya dapat menggunakan *key* di dalamnya tetapi tidak dapat mengekstrak apa isi dari *key* tersebut.

2. *Key* yang diletakkan di *secure hardware*

   Ketika fitur ini diaktifkan untuk sebuah *key, key* tidak akan keluar dari elemen *Secure hardware* (contoh: *Trusted Execution Environment* (TEE) dan *Secure Element* (SE)). Ketika penyerang mendapati Android OS dan membaca media penyimpanan internal, penyerang hanya dapat menggunakan *key* dari *keystore* aplikasi apa pun tetapi tidak dapat mengekstraksinya dari *keystore* tersebut. Fitur ini hanya dapat bekerja jika elemen *secure hardware* perangkat Android mendukung kombinasi tertentu antara *key algorithm, block modes, padding schemes*, dan *digests* terhadap *key* yang digunakan. Untuk mengecek jika fitur ini dapat aktif untuk suatu *key*, dapat mengecek *return value* dari fungsi **KeyInfo.isInsideSecurityHardware().**

2.4.2. Key Use Authorizations

Untuk mencegah penggunaan *key* yang tidak diotorisasi oleh perangkat Android, Android Keystore membuat aplikasi menspesifikasikan penggunaan terotorisasi *key* nya ketika proses generasi dan import. Setelah proses generasi atau import, otorisasinya tidak dapat diubah lagi. Proses otorisasi kemudian dilakukan oleh Android Keystore setiap kali *key* digunakan. Otoritas penggunaan kunci yang didukung termasuk dalam kategori berikut [14].

1. *Cryptography*

   Authorized key algorithm, operasi (enkripsi, dekripsi, *sign*, verifikasi), *padding schemes, block modes, digests* di mana *key* dapat digunakan

2. *Temporal validity interval*

   interval waktu hingga *key* terotorisasi dapat digunakan

3. *User authentication*

   Key hanya dapat digunakan pengguna jika pengguna telah terotentikasi.

2.5. QR Code

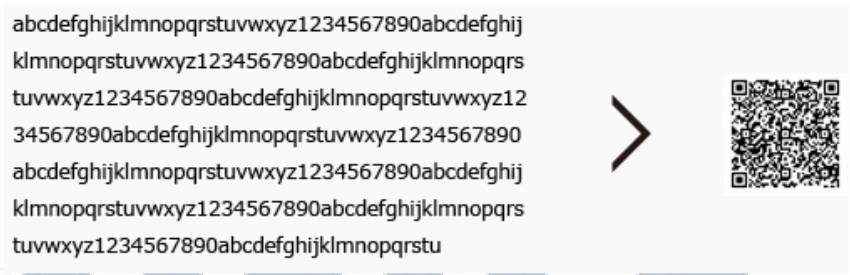
QR code (Quick Response Code) pertama kali dikembangkan oleh DENSO WAVE untuk keperluan industri otomotif. QR code dapat menyimpan data dengan empat metode encoding yang standar yaitu (numerik, alfanumerik, byte/biner, dan kanji). QR code menjadi sangat populer di luar industri otomotif dikarenakan keterbacaan yang cepat dan kapasitas penyimpanan data yang lebih besar dibanding UPC barcodes.

QR code terdiri dari persegi hitam yang tersusun pada grid persegi yang berwawrna putih yang dapat dibaca dengan perangkat pemroses gambar seperti kamera, dan diproses menggunakan Reed-solomon error correction hingga citra dapat diinterpretasikan secara tepat. Data yang dibutuhkan kemudian diekstrak dari komponen horizontal dan vertikal pada citra.

QR code dapat menyediakan beberapa fitur yang diunggulkan yaitu [15].

1. *High capacity encoding data*

   UPC Barcodes yang konvensional hanya dapat menampung maksimum 20 digit angka. QR code dapat menampung belasan hingga ratusan kali lipat dari jumlah tersebut. QR code juga dapat mengatasi berbagai tipe data seperti numerik, alfanumerik, simbol, kanji, kana, hiragana, dan biner. Satu simbol QR code dapat menampung hingga 7089 karakter.



Gambar 2.3 Kapasitas QR code [15]

2. Ukuran *print out* yang kecil

   Dikarenakan QR code menampung data secara vertikal dan horizontal, QR code dapat menampung data yang sama dengan UPC Barcode dengan ukuran yang lebih kecil 10 kali lipat.

3. Kemampuan menyimpan karakter kanji dan kana
4. Resistensi terhadap debu dan kerusakan

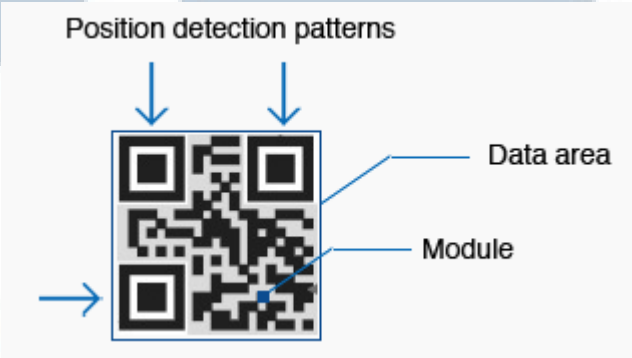
   QR code memiliki kapabilitas *error correction*. Data bisa dikembalikan seperti semula walaupun QR code rusak atau tidak terlihat sebagian. Codeword

adalah unit yang mengonstruksi data pada sebuah area. Pada QR code, satu codeword sama dengan 8 bit. Maksimum codeword yang dapat dikembalikan adalah 30%.

Gambar 2.4 Kapabilitas QR code mengatasi debu dan kerusakan [15]

5. Dapat dibaca dari 360 derajat

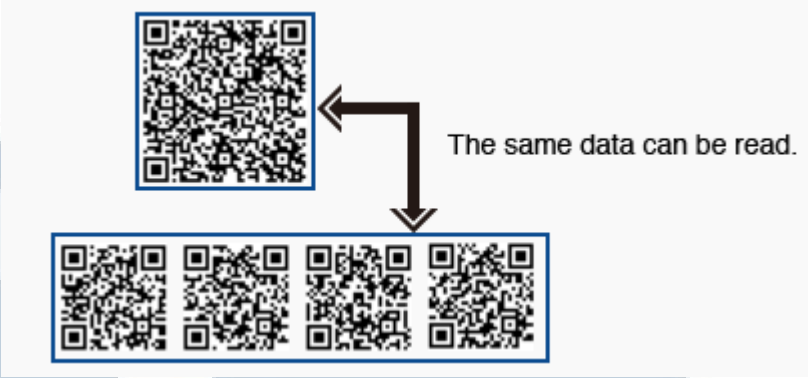
QR code mampu dibaca dari 360 derajat. Fitur ini dapat dicapai berkat pendeteksi pola posisi yang terdapat di tiga sudut pada QR code. Tiga pendeteksi pola posisi di sudut ini menjamin pembacaan yang cepat dan stabil.



Gambar 2.5 Pendeteksi pola posisi [15]

6. *Structured appending feature*

QR code dapat dibagi menjadi beberapa area dan sebaliknya, beberapa QR code dapat digabung menjadi satu QR code. Satu data simbol dapat dibagi menjadi 16 simbol, sehingga dapat dicetak pada area yang lebih sempit.



Gambar 2.6 Structured appending feature [15]

2.6. Cip NFC (NTAG 215)

Pada setiap tag NFC (*Near Field Communication)*, terdapat cip NFC yang berupa perangkat elektronik kecil untuk menyimpan data dan mengontrol bagaimana cara data tersebut diakses. Cip NFC tersedia dalam berbagai kapasitas memori dan konfigurasi memori yang berbeda yang berpengaruh pada seberapa banyak informasi yang bisa disimpan oleh cip dan bagaimana cip tersebut dapat dikunci.

NTAG (NFC Tag) dirancang berdasarkan spesifikasi standar industri ISO/IEC 14443 dan diproduksi oleh NXP Semiconductors. Pemroduksi cip NFC terbesar adalah NXP Semiconductors dan Broadcom. Ada banyak jenis varian cip NFC yang terkenal dan sering digunakan seperti NTAG213, Mifare Ultralight, Mifare Classic 1k, Topaz 512. Berikut spesifikasi beberapa NFC Tag yang populer:

	Mifare Ultralight	Mifare Classic 1k	Mifare DESFire 4k	NTAG 213	NTAG 215	NTAG 216
Memori Total (byte)	64	1024	4k	180	540	924
Memori Pengguna (byte)	48	716	4094	144	504	888
Universal compatibility	Ya	Tidak	Ya	Ya	Ya	Ya
NFC Forum Compliant	Ya	Tidak	Ya	Ya	Ya	Ya
Serial Number	7 *bytes*	4 atau 7 *bytes*	7 *bytes*	7 *bytes*	7 *bytes*	*7 bytes*
Kriptografi	Tidak	Ya	Ya	Tidak	Tidak	Tidak
Kekuatan baca	Tinggi	Sedang	Sedang	Sangat tinggi	Tinggi	Tinggi

Tabel 2.1 Tabel perbandingan jenis cip NFC populer [16]

Dari tabel di atas, penjelasan setiap keterangan adalah sebagai berikut:

1. Memori Total

   Total ukuran memori pada cip NFC. Mode pembacaan dan penulisan memori dapat berupa OTP (*one time programmable*), pembacaan dan penulisan diatur berdasarkan *locking features*, dan kebanyakan tidak diatur sama sekali.

2. Memori Pengguna

   Kapasitas memori yang dapat digunakan untuk menyimpan data.

3. *Universal Compatibility*

   Universal compatibility berarti dapat digunakan dengan seluruh ponsel dengan kapabilitas NFC.

4. *NFC Forum Compliant*

   Cip NFC dirancang berdasarkan spesifikasi NFC forum.

5. Serial Number

   Cip memiliki nomor serial yang unik guna untuk proses identifikasi.

6. Kriptografi

   Fitur keamanan pada cip yang dapat mengamankan cip dari kloning dan akses data yang tak terotorisasi.

7. Kekuatan Baca

   Kekuatan baca rata-rata cip NFC. Fitur ini terkait dengan jarak seberapa dekat cip harus berada jika akan dibaca.

2.7. Penelitian Terkait

Ada beberapa penelitian terkait penelitian kali ini. Beberapa penelitian telah mengkaji pembuatan protokol pengganti kata sandi yang notabene buruk bagi pengguna dan dari sisi keamanan. Ada dua protokol yang mencoba menggantikan kata sandi sama sekali dan

2.7.1. FIDO *U2F*

Protokol FIDO *U2F* (*universal second factor)* membuat proses *log in* ke situs menjadi lebih mudah bagi pengguna yaitu hanya dengan cara mempresentasikan perangkat faktor kedua. Dengan dibutuhkannya perangkat tersebut pengguna dimungkinkan untuk menggunakan kata sandi yang sederhana seperti 4 digit PIN dan tetap aman.

Secara sederhana, proses kerja FIDO *U2F* melibatkan proses registrasi dan otentikasi. Pengguna cukup mempressentasikan perangkat

faktor kedua ke komputer lewat USB, dan pengguna cukup memverifikasi keberadaan dengan cara menekan tombol. Pengguna dapat menggunakan perangkat FIDO *U2F* untuk banyak situs. Namun, penggunaan perangkat khusus FIDO *U2F* ini hanya bisa bersamaan dengan browser yang telah mendukung FIDO *U2F*.

Token *U2F* menyediakan kemampuan kriptografik yang dapat diverifikasi oleh server dan kriptografi. Token umumnya digunakan sebagai faktor kedua (setelah kata sandi) saat proses otentikasi. Token *U2F* berukuran relatif kecil dan berupa perangkat dengan fungsi khusus yang tidak terkoneksi dengan internet tetapi dapat berkomunikasi dengan server. Maka dari itu, token FIDO *U2F* memerlukan FIDO *client* untuk me-*relay* pesan antara token dan server. Pada umumnya FIDO *client* berupa web browser.

Protokol FIDO *U2F* dibagi menjadi dua proses besar yaitu proses registrasi dan proses otentikasi. Proses registrasi akan mendaftarkan pasangan kunci (*key pair)* yang telah diregistrasikan sebelumnya ke server. Kedua proses tersebut terdiri dari 3 fase yaitu:

1. Setup

   Pada fase ini, FIDO client akan mendapatkan *challenge* dari server. Dengan *challenge* tersebut, FIDO *client* akan membuat *request message* kepada token *U2F*.

2. Processing

   Pada fase ini, FIDO *client* akan mengirim request pesan ke token, dan token akan melakukan operasi kriptogafi pada pesan tersebut dan membuat *response message*. *Response message*ini kemudian akan dikirim kembali ke FIDO *client*.

3. Verification

   Pada fase ini, FIDO *client* mengirim *response message* dari token bersamaan dengan data-data lainnya yang diperlukan oleh server untuk melakukan verifikasi token *response* ke server. Server kemudian akan memroses *token response* ini untuk membuktikan kebenarannya. Jika benar, maka pada proses registrasi, server akan mendaftarkan *public key* yang baru untuk usesr tersebut sedangkan jika pada proses otentikasi,

maka server akan menerima bahwa klien memiliki *private key* yang dibutuhkan dan dapat melanjutkan ke proses selanjutnya.

2.7.2. Pico

Pico adalah suatu protokol otentikasi yang dirancang oleh tim Frank Stajano di *Cambridge University*. Tujuan utama dari dirancangnya Pico adalah untuk merancang metode otentikasi tanpa harus mengingat kata sandi yang sulit ditebak (juga biasanya relatif sulit diingat). Pico dirancang agar setidaknya dapat memiliki fitur yang lebih dari kata sandi pada hal berikut ini [17]:

1. Memoryless

   Pengguna tidak perlu mengingat rahasia / kata sandi apa pun.

2. Scalable

   Sistem otentikasi harus dapat digunakan bahkan pada beberapa situs dan aplikasi yang berbeda-beda.

3. Secure

   Sistem otentikasi baru ini setidaknya harus seaman sistem kata sandi.

Selain fitur-fitur tersebut, sistem Pico ini berupa sistem otentikasi berbasis token yang menjadi faktor penentu otentikasi. Oleh karena itu, sistem Pico juga dirancang dengan memikirkan faktor berikut [17]:

1. Loss Resistant

   Jika token tersebut hilang atau rusak, pengguna harus dapat mendapatkan kembali akses ke dalam sistem.

2. Theft Resistant

   Jika token dicuri, pencuri harus tidak dapat masuk ke dalam sistem bahkan jika diasumsikan pencuri dapat menyadap token dan mengakses sistem di dalamnya.

Jika dilihat dari sisi kebergunaan, Pico dirancang supaya memenuhi syarat kebergunaan berikut [17]:

1. Works For All

   Dapat digunakan tidak hanya untuk otentikasi pada web tetapi juga aplikasi

2. From Anywhere

Pengguna harus dapat mengotentikasi tidak hanya dari web tetapi juga berbagai *client*.

3. No Search

   Pengguna tidak perlu memilih kredensial mana yang ingin digunakan

4. No Typing

   Pengguna tidak perlu mengetikkan lagi kata sandi dalam bentuk apa pun

5. Continuous

   Otentikasi bersifat berkelanjutan, tidak hanya pada saat *session start*

Pico juga dirancang untuk memenuhi standar pada bidang keamanan dan harus memiliki ketahanan serangan sebagai berikut [17]:

1. No weak

   Pengguna tidak menggunakan kata sandi yang lemah

2. No reuse

   Pengguna tidak dapat menggunakan kredensialnya untuk situs / aplikasi lain

3. No phishing

   Phishing tidak dapat terjadi

4. No eavesdropping

   Penyadapan melalui jaringan tidak dapat terjadi

5. No keylogging

   Keylogging tidak dapat terjadi

6. No surfing

   Shoulder surfing tidak dapat dilakukan

7. No linkage

   Dua token yang berbeda tidak dapat dihubungkan (harus memiliki kredensial yang berbeda).

Pico didesain sehingga berukuran kecil, *portable*, perangkat khusus, dan diperuntukkan agar selalu dibawa ke mana-mana seperti jam tangan dan kunci rumah. Perangkat pico memiliki dua tombol penting yaitu “main” dan “pairing”, layar kecil, kamera yang mampu untuk mengambil gambar kode visual 2 dimensi, dan antena radio bidireksional berjarak pendek [17].

Secara internal, memori permanen Pico (terenkripsi) dan memiliki ribuan slot yang satu slotnya berfungsi untuk melakukan *pairing* antara aplikasi dan pico. Setiap slot memiliki kredensial berupa *private key* dan hal lainnya yang diperlukan untuk proses otentikasi [17].

Aplikasi umumnya akan menampilkan kotak untuk penginputan userid dan kata sandi. Pada implementasi pico, *request* akan tertanam pada kode visual 2 dimensi dan ketika visual tersebut sedang muncul pengguna dapat melakukan salah satu dari dua pilihan aksi yaitu melakukan pembacaan kode visual 2 dimensi dengan kamera yang terdapat pada perangkat pico atau menekan tombol "main" untuk mengirimkan kredensial yang telah ditetapkan sebelumnya (seperti kata sandi) atau tombol "pairing" jika sedang dalam proses pembuatan akun baru [17].

Banyak dari sistem token based yang melupakan kelemahan jika token tersebut dicuri. Contoh sederhana adalah kunci mobil. Siapa saja dengan kunci mobil dapat memiliki akses terhadap mobil. Ada pengamanan pada token yang umumnya berupa PIN atau kata sandi singkat. Tetapi hal tersebut membuat masalah mengingat pada manusia kembali lagi. Sedangkan hal yang ingin dicapai pico adalah *memory less*. Pada sistem pico, pico menggunakan picosiblings untuk mengunci pico jika pico tidak berada di dekat picosiblings. Picosiblings harus ada di dekat pico jika ingin melakukan otentikasi dan picosiblings dapat berjumlah lebih dari satu. Kredensial pada pico akan terus terenkripsi dan hanya dapat didekripsi melalui picosiblings. Pico dirancang sebagai alat yang perlu dibawa, sedangkan picosiblings akan dibuat lebih dekat lagi dengan pengguna seperti ditanam pada ikat pinggang, dompet, dan perhiasan. Komunikasi pico dan picosiblings menggunakan protokol yang ada pada protokol "Resurrecting Duckling". Protokol ini menjaga koneksi dengan menggunakan skema komunikasi ping dan pong untuk bertukar data seperti berikut:

$$\text{Ping for Picosibling } j, \text{broadcast by Pico}: \quad EtMAC_{k_{3,j}}(\texttt{"ping"}, r, c_j)$$

$$\text{Pong from Picosibling } j: \quad EtMAC_{k_{3,j}}(\texttt{"pong"}, r, c_j, s_j)$$

$$\text{Both (with appropriate sync)}: k_{3,j} := h(k_{3,j})$$

Gambar 2.7 Skema ping pong pada pico dan picosiblings [18]

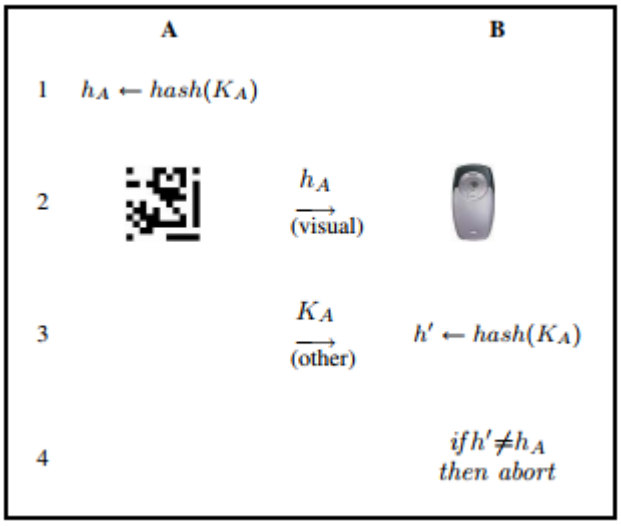
Pico dan picosiblings *j* mengetahui satu *symmetric key k3* yang sama yang kemudian akan digunakan untuk *Encrypt then MAC* pada kedua ping dan pong. Namun, setiap kali dilakukan ping dan pong, *symmetric key* ini akan diperbarui dengan melakukan hash. Dengan cara ini, maka walaupun suatu saat penyerang dapat mendapatkan *symmetric key* tersebut, penyerang tidak dapat membaca pesan yang lama. Counter *Cj* juga termasuk dalam teks disamping random nonce *r* [18].

2.7.3. Seeing is Believing Protocol

Protokol ini dikembangkan oleh Jonathan M. McCune dkk pada tahun 2005. Protokol *Seeing is Believing* (SiB) menggunakan *barcode* 2 dimensi (contoh: Data Matrix, PDF417, Maxi Code, atau QR code) dan ponsel berkamera untuk mengimplementasikan teknik otentikasi menggunakan *channel* visual. Ketika pengguna melakukan otentikasi dengan protokol SiB, pengguna harus mengarahkan kamera ponselnya ke arah *visual 2D barcode*. Aplikasi ponsel kemudian akan membaca *visual 2D barcode* dengan *barcode recognition algorithm* untuk memroses gambar tadi menjadi sebuah informasi [19].

Terdapat dua buah protokol utama pada protokol SiB yaitu *bidirectional authentication* dan *unidirectional authentication*. Pada protokol otentikasi *bidirectional*, kedua belah pihak akan melakukan pertukaran *public key* kemudian kedua belah pihak harus melakukan *scan 2D barcode* untuk memverifikasi bahwa *public key* yang diterima itu adalah benar [19].

Pada protokol otentikasi *unidirectional*, salah satu *device* adalah pasif dan ter-embed *key pair* yang sama seterusnya. Pengguna yang ingin mengotentikasi dirinya ke device tersebut akan mendapatkan *device public key* dan diverifikasi menggunakan *scan 2D barcode*. Ada pun skema pemverifikasian protokol SiB sebagai berikut (K_A adalah Public Key milik A) [19] :



Gambar 2.8 Protokol pre-Otentikasi SiB [19]