QwyitCard™ and EMV Fraud

Introduction, Reference Guide and Discussion

Version 1.0 June, 2018

Abstract

This paper provides an introduction to the problems of Credit Card fraud. The current, costly EMV smart chip cards are discussed, and issues presented. To address these issues, the QwyitCard™ solution is proposed. The Reference Guide details are given, including the new Customer Verification Value called a Proof Key and its associated Challenge/Response are then presented. Summaries of the process, the Card improvements, the operational code and the security are included; along with a discussion of the business advantages in adoption and proliferation of the QwyitCard™.

Contents

QwyitCard – The EMV Credit Card Fraud Solution

Part I – The Problem

You've probably received new smart chip credit cards over the last few years to replace your old ones. These are called EMV Chip and Signature (or the less used in the US, Chip and PIN) 'smart cards', introduced by Europay, Mastercard and Visa. These are for fraud reduction introducing two main characteristics: they can deliver a unique identifier for each transaction along w/the static card number providing more security, and they store and contain encrypted data making the cards difficult to counterfeit. Awesome!

Over 20 years ago, in the early days of my cryptographic adventures, I had an idea for 'perfectly secure Credit Card transactions' called Armour Card. This was to send a one-time-pad version of the card number whenever submitted digitally (Internet and/or through a card reader) using Qwyit® cryptographic primitives and a new number imprinted on the unused 3rd magnetic stripe (still unused today!) I called this an Electronic Commerce Number (ECN). I covered the counterfeiting problem by simply mixing that ECN with a 4-digit hexadecimal PIN that folks would easily remember and enter upon every use. Yes, I was ahead of my time, since those are the exact same two reasons for the EMV 'smart chip' technology.

[My version was (and still is) unbreakable, because my process provides a One-Time-Pad *version* of the Credit Card Number (CCN) – this is superior to the current smart cards, in that they provide a one-time identifier, but it is *not* integrated with the CCN…just in addition to it. And the way that ID is formed is *not unbreakable* – we'll get to all that in a minute!].

I somehow scored an executive-level meeting w/Chase Manhattan Bank (their name then) in 2000, when they put out a call for new-fangled electronic techniques for financial transactions. I had a 12-page deck, introducing the 3rd stripe ECN on the first slide. Several executives all up and down a huge conference table…I start – say a few sentences, introduce the ECN on the unused 3rd mag stripe – and I get interrupted:

"You can't touch the card," someone says from the table.

Now…I'm dead in the water. I try to say a few more sentences about security, it's important…and when I mention that I read their annual report and that their credit card fraud rate was over 4%...another interruption:

"That's in the noise. We don't care about that."…Whoa. Needless to say, the rest of the meeting was…short.

Let's jump ahead to 2015. That 4% turns into ~$3*billion* in US retail merchant market losses. And the solution? A ~$3*billion* implementation of those EMV smart card chips – which is only about 60% complete – estimates for a total EMV solution rollout are…***$8-12BILLION****!* And the fraud rate is only down 58% from March 2016 to March 2017.

H'mmm…I don't know about you, but that bothers me. First, we have an actual credit card company limit the possibilities; then we have them turn to current cryptography methods to provide an answer to their no-longer-small problems; instead of how they originally asked for innovative approaches. The result is an extremely costly solution that only works somewhat – because of the cost, the total expansive effort to replace the entire infrastructure, and the processing time that antagonizes the user – oh…and the cards can still be hacked (more on that later). I had a perfect solution over 20 years before that *works 100%* - with no new costs whatsoever:

- The same credit cards can be used – no additional cost to print an ECN on the unused 3rd mag stripe
- The cards offer 100% protection against fraudulent use *even when lost or stolen* – they are PIN protected, with no PIN storage on the card

- The same readers can be used – just a quick firmware update to perform the simple, fast, small easy calculations
- The end user experience is identical to then-debit cards, and current Chip and PIN EMV cards
- The end user experience is enhanced as the card-related processing time (exclusive of bank approval time) would be less than 100 milliseconds as opposed to the current over 4 seconds for average EMV smart chip card transactions (this is greater than 3 orders of magnitude faster)
- The actual data transaction transmission is 100% provably unbreakable, meeting the OTP requirements

All of this, over 20 years ago…for zero cost. Its *$3BILLION* today and counting up to *$8-12BILLION*…Oh My.

Part II – The Current EMV Method

When most people hear me talk about the above, they nod in agreement. But the bottom line is, almost no one actually believes it is true. Not that they think I'm lying; but more along the lines that 'my tech solution' can't really be that simple, complete – or actually work. After all, besides the above not being a technical description, the folks who came up with the current methodology – in this case EMVCo, which is Europay, Mastercard, Visa, etc. – are some really Big Dogs! These people know what they're doing! And you can go vouch for the complete mastery of the messaging exchanges that are 'required' for a comprehensive, secure credit transaction by reading the current EMV 3DS 2.1.0 Specification. Whoa – now *that's some serious tech!*

But the first thing to realize about this is: if you want to know whether The Status Quo of a situation is the best overall solution, you need to start at the beginning. And that, was in Part I above:

- There is a great deal of credit card fraud
- This comes from unauthorized use of the cards
- This comes from stealing/losing the cards
- This also comes from the static nature of the transaction (same Credit Card Number (CCN), etc.)

So any 'tech' or 'method' proposed is – should be – all about solving this fraud problem. *You must remember that is the entire reason for that complex, technology spec – and for putting a 'computer' on the card so it isn't 'static' anymore*. It doesn't feel like there can be a simple, straightforward, *much less complex* answer than that spec. When these folks began talking about it maybe 8-10 years ago, they were at the same exact place I was 10 years before that when I proposed my solution. But instead of starting from scratch, removing *all of their current concepts, strategies, partial solutions to different parts of the problem (as they saw it), and all existing methods,* ***they took what they already did and added more stuff***.

What this does, is make assumptions that what they were already doing solved the 'initial problems' that existed in order to 'just do credit card transactions.' And that those *are still there, therefore the solutions they had must be kept!* And from there, they would 'define the new parts of the problem' that are leading to all this fraud, and bolt new stuff on/around/in the existing methods to 'solve those'…and then that gives us a Total Solution!

Hopefully, you can see the problem – it's exactly why Rube Goldberg offered up his 'contraptions': this *isn't the way to do it and it can plainly be shown by going all the way to extremes.* And that is *exactly* where we are today: $*12Billion* being pushed into that spec…down the throats of merchants who can't possibly understand it…but they are left with no choice but to obey it.

And besides convoluted technology, what else did EMV do to help solve the fraud problem?

Here's what they did: they reversed the liability of fraudulent charges. So not only have they mandated a difficult, *egregiously expensive technology solution*: they decided that what they've done is **so good**, they make any merchant who doesn't buy it, *pay for it anyway* – because any non-abiding merchant will be stuck with the cost of 'existing' fraud using 'the old doesn't-work' technology. Whoa…*again!*

But…is there something else that could solve the problem? We're back to my presented idea…hopefully, with you a little bit further toward acceptance of it because you understand what has led us to where we are:

- The bank ignored – and obstinately blocked – original calls for complete credit transaction processing ideas/methods/protocols until it was way too late
- When they finally got around to proposing their solution, there was absolutely no consideration given to completely new methods/approaches, just Rube-like extensions of what they already did
- There was absolutely no consideration given to the mind-exploding cost of their methods
- They doubled-down on their tragic solution by switching the decades-old liability forcing merchants to lose either way: use and pay or don't use and pay too!
- *In summary, all of what transpired was for the benefit of the creditors, and at the expense of the merchants and consumers*

Oh…and remember how I mentioned previously, that after framing the history of how we got to where we are today, I'd get to how their EMV solution actually works?…Well – here's the end result:

Without wasting any ink on all the various technical scenarios for how EMV and/or mag stripes *attempt* to solve card fraud, here's the summary:

> *According to Aite's 2016 Global Consumer Card Fraud report, it is safe to assume that all users have been compromised. Whether you use a card with a magnetic stripe or a more secure chip-and-PIN card doesn't matter — if you have a card, its information has probably been stolen. Now that criminals have developed a method to actually clone the cards, that starts to look like a very serious financial threat. – from Kaspersky Daily, 3/9/2018*

H'mmm. So…*All of the cards in use today, all of the methods employed – Chips, PINs, CVVs,* ***every method they use:***

Do Not Work.

Additionally, they don't work because even when they have a method, they don't do it! The hack from the above story using Chip and PIN cards was because the customer verification (CV) step isn't mandatory in EMV – and even when they do it, they try to do it between the card reader and the card using CVV/CVV2/PIN, *not all the way to the Issuer!* This is because of the exact same problem(s) that exist in all current security communications: *the methods are so complex they can't be done in real time!* If all those Chip and PIN cards actually communicated w/the Issuer for CV, you'd have to stand at the terminal for an eternity – and it *still wouldn't use an unbreakable transaction methodology!*

Since this isn't an 'acceptable customer experience', and the merchants will be framed for all the problems anyway:

Fraud has been chosen as the 'solution' to getting you to continue using the cards.

At $3*Billion and counting to $12Billion…*

EMV cards don't deliver what they promise – their technology solution is a fraud. EMV commits further fraud by forcing a known, broken solution onto all of their customers – and makes them liable for their failings.

I was upset 20 years ago when I presented. I was upset 6 years ago when I saw they 'touched the card'. I was upset 3 years ago when I saw what it was costing merchants – including some of my friends and business associates. And I'm upset now as I write this. The only thing that would help me, is if someone reads this who matters in this arena…and they get upset too!

Part III – The QwyitCard™ Method (QC™) – A Reference Guide To Implementation

The purpose thus far was to demonstrate how the Status Quo of cryptography/security has been in concert with established business entities intent on protection for their (extremely valuable) domains with a complete disregard for innovation, cost savings, customer protection, and actually to the detriment and severe financial consequences of all involved except themselves.

But there is a simple solution to all of these problems, and it exists within the Qwyit® authentication and data security technology proposed over 22 years ago. In order to articulate *exactly how that technology solves those problems*, the following QwyitCard™ Reference is presented:

Here's *the complete, exact Qwyit processing to 100% securely (unbreakably) transmit a static CCN to the proper Credit Authorizing agent and receive approval/denial with 100% secure (unbreakable) confidence for the Merchant – using OLD magnetic stripe cards for no additional cost.*

The process is as simple as it can be:

Requirements

- Each creditor will send authorized customers a new Credit Card, with the pertinent information for credit requests (Name, etc. – as per current criteria)
    - The card includes the familiar 16-decimal digit Credit Card Number (CCN) and a 16-hexidecimal digit Electronic Commerce/Card Number (ECN) stored on the unused 3rd magnetic stripe on the card back
        - The ECN is a random hex number (0-F), with no special requirements
        - The ECN is offset with a PIN, a 4-hex digit number that is given to the customer along w/the card
            - The PIN is integral *to the card* – not to *you*, as you've been led to previously believe about PINs (probably because of their silly misnomer!). They should really be thought of more like a 'Product ID' or 'Place ID'…more like the "ID" you create when you use home security (when you call the alarm company, there's a 'passcode' you give them to identify your home – anyone there can use it), or like when you open a wireless account, or an ISP (like FIOS, Xfinity, etc.) – the QC™ PIN is just like that call-in identifier. You can't use the card w/o knowing the PIN – and whoever knows the PIN is using *its card*
    - The card may/may not include today's 'industry standard Card Verification Value', CVV/CVV2
        - Whether this is utilized isn't necessary – See Appendix D for information
    - The card includes a *Proof Key* (PK), which is an 11-decimal digit number with space between the values on the card back
        - See Appendix D for the complete PK description – the value is *not* available on the mag stripes
        - PK is a random arrangement of the digits 1-9 non-repeating, bookended by the digit 0 on both ends [There are 9! (362,880) possible arrangements]
            - Example: 0 3 9 8 1 2 5 7 4 6 0

This new credit card is a called a ***QwyitCard™*** (QC™). (We can still call it the *Armour Card*, if anyone insists!)

[*Note*: Since it is standard operating procedure for Card companies to mail out new cards, with associated PINs, this is a simple requirement to meet. The only difference will be that the PIN is *integral* to the card – if the user wants to change it, they'll have to go to a location to re-stamp their current card, or be sent a new one; or not be allowed to change it!]

- Each Merchant will possess a QC™ reader capable of performing QC™ Authorization (QCA) processing

- Each creditor, along with any other allowed/authorized transaction processing participants, will also be capable of performing QCA processing

Process

- **Step 1:** The QwyitCard™ is inserted into a QC™ reader, accessing the CCN, the ECN, along w/any other open identifying information (Name, an Open ID number of some kind, etc.) needed for the authorization submission

  Note: Nothing in the card (recorded on the mag stripes) can be used if it is lost or stolen – even the CCN! This is because there is no way to submit *just the CCN* for authorization. The remainder of this QC™ process *must be completed in order to use the card!* The process will result in a one-time-only unique 'encrypted' CCN that is submitted to the authorizing agent. This encrypted CCN is called the CCN OTP from the One Time Pad encryption algorithm that is the only provably unbreakable cipher. The following steps in this process create exactly that: a OTP version of the CCN that is unbreakably unique. Remember, the stored ECN on the card is 'encrypted' (offset) using the PIN – without it, the process can't be completed correctly!

  Note: The entirety of 'Card Not Present' (CNP) transactions are discussed in full in Appendix D, Proof Key Challenge/Response – this process outline covers mag stripe swiped transactions.

- **Step 2:** The QC™ reader asks the user to enter their PIN

- **Step 3:** Next, a new random number is created – an open, non-secret number called the OpenR. It is a 16-hex digit value

- **Step 4:** The PIN is added to the retrieved ECN to 'unlock' it – the stored version needs that PIN to work.

  Note: In this QC™ process, the meaning of 'added' is that of modular arithmetic: where you add the digits together, and if the sum is larger than the system base (in a hexadecimal system, this is 16, since there are 16 unique digits), then you drop that value, and use the remainder as the result. I.e., if you add a '9' with a 'C', you get a '5'. This is because a 'C' is already a '12' and adding '9' gets us to '21'. So we drop the first '16', and are left with '5'. Right?! In QC™ adding, all of the values return values of the same number of digits, after dropping the base where needed. We have a Qwyit® primitive function we call MOD16 that does this for all the adding. (And MOD16D for decrypting, which finds the missing input number when given the other input number and the result)

- **Step 5:** The OpenR is added to the decrypted ECN to create an offset called the OpenRMixed

- **Step 6:** Then one of the Qwyit® protocol functions called the *Position Digit Algebra Function* (PDAF) is performed using the OpenRMixed value and the decrypted ECN

Note: Now this is fun! The PDAF is a combination of pointing and adding (hence the position and algebra in the title!) The quick skinny on how it works is this: You take two values, the first one is the 'Key', and the second one is the 'Pointer'. You take the 1st digit of the Key and add it to where the 1st digit of the Pointer points, when aimed back at the Key. When you do this, you point at a position that the Pointer-value indicates *to the right* of where you are in the Key, with the 1st digit to the right being the 0th position. So if the Pointer-value is a '2', you will go *three* digits to the right of where you are (we started at the 1st Key digit) and add that Key digit's value to the value where you started. For instance, if the Key is '1234' and the Pointer is '215F', you would add the '1' from the Key (its 1st digit value) to the '4', which is chosen because the Pointer's 1st digit value is a '2', so we select the 3rd Key digit value to the right of where we were, to add. Right?!

This isn't as complicated as it sounds…just re-read that a couple times if you didn't get it – because it's an amazingly powerful tool. Not only is it just algebra so it is blazingly fast, but…the PDAF can be configured to use the Pointer's values to add to the Keys, you can switch them, you can offset where you start, you can jump over values (called the Skip Setting), Oh MY! And all of *those options can be 'chosen randomly' by the digit values themselves!* That means that the PDAF can function differently, pseudo-randomly *and automatically!* Now you know why in the step right before this one, we 'mixed' the OpenR – so there is no knowledge available of *where* the OpenR is pointing into the ECN (remember, the OpenR is a non-secret value). It really isn't necessary, as that knowledge won't help anyone solve these underdetermined equations, but…since it takes *no time at all, why not?!*

If you got this, great…if not – don't worry. All you need to know is: *there is an incredibly powerful, extremely fast way to combine numbers that give unique, unsolvable results without any of the madness of the current Smart Chip methods: they are insanely complicated, overly computationally expensive, and they don't provide the perfect results of a PDAF*.

- **Step 7:** Next, we'll perform another Qwyit® primitive called the *One Way Cut* (OWC) to halve the PDAF result

  Note: We used the PDAF to expand the resulting values from 16 to 38 digits (by going through the Key and Pointer twice plus 6 digits of a 3rd round, just shifting the start of the second and 3rd rounds to the 2nd digit of both numbers, then the 3rd digit of both). We did this to set up another perfect, one-way gate: since the PDAF result is now a double-length value (plus 6 additional digits we'll use in the next step), we'll cut it back in half to 19 digits using the OWC. This function simply adds up the digit pairs of the input number, resulting in a value that is half the length. This function is another perfect, permanent one-way gate: you simply can't know the two digit values that were added to get one digit! Can you tell me what two numbers I added to get a '3'?! The 1st 16 digits of the OWC result are called the CCN Offset; the last 3 digits are called the PK Offset – and are used in the next step.

- **Step 8:** There is a wonderful 'Customer Verification Value' (CVV) mechanism included in this QC™ technique called the Proof Key Challenge/Response (PK C/R) – which is explained in detail in Appendix D. The PK OTP is created by simple addition of the PK C/R asked/answered info with those last 3-digits of the preceding OWC. This result is a unique, one-time only value based on the per-transaction generated OpenR and the real PK printed on the card

- **Step 9:** Lastly, the CCN OTP is created by simply adding the real CCN on the card to the CCN Offset. This result is a unique, one-time only value based on the per-transaction generated OpenR and the real CCN

- **Step 10:** The QC™ reader then sends the cardholders Name/OpenID, the OpenR, the unbreakably encrypted CCN OTP and PK OTP, along with any other required info to the Authorizing Agent for approval/denial

That's it – This simple 10-step process from swipe to send is nothing more than what is called an 'underdetermined linear equation set' – 'underdetermined' means there are more unknown variables than there are equations to solve them, so they are *unsolvable: unbreakable*. 'Linear' means they are simple addition – and *nothing is faster than that!*

The QwyitCard™ process takes the static card values, generates a single non-secret value, and by insolvably linear mixing of these, then delivers a perfect, unbreakable *version* of the CCN to the Authorizing agent that can *only have been presented by the PIN-knowledgeable owner of the card – and it is unique every single time!* And it does this with mind-numbing speed and almost no computing power required. For *$Zero* – that is *for no cost whatsoever* to the end-user, the merchants, and even the creditors. Without new chips…new readers…!

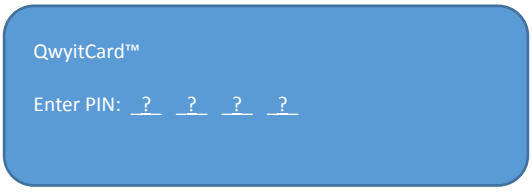
Here it is in pictures:

Step 1 – Swipe Card

You have received your new QwyitCard™ in the mail…with PIN 8F97. The front looks just like it always did…

The back is a bit different. It has the CCV/CCV2 (maybe). The ECN is on the magnetic strip at the top (you would not be able to see it, shown for emphasis). There is a new PK number.

To purchase, you just swipe like you always did!



Step 2 – Enter PIN

The QC™ card reader asks you to Enter your PIN:

You type in: 8F97

Random Number Generator is used to create a number:
2F1C1C1D31867384

Step 3 – Generate OpenR

The QC™ reader generates a non-secret 16-hex digit value, called the OpenR:
Result = 2F1C1C1D31867384

QwyitCard™

3ECBE1954BCC35D4
+ 8F978F978F978F97
BD52602CCA53B46B

Step 4 – Unlock ECN

The QC™ reader performs a MOD16 add of the PIN with the ECN, decrypting it (ECN_d):
Result = BD52602CCA53B46B

QwyitCard™

2F1C1C1D31867384
+ BD52602CCA53B46B
DC6E7C39FBD927EF

Step 5 – Mix the OpenR

The QC™ reader adds the OpenR to the ECN_d to mix it, creating the OpenRMixed:
Result = DC6E7C39FBD927EF

QwyitCard™

DC6E7C39FBD927EF
+ BD52602CCA53B468
13F715798A1364A668A4A2516CF566C66888C6

Step 6 – PDAF

The QC™ reader performs a PDAF using the OpenRMixed to point into the ECN_d:
Result = 13F715798A1364A668A4A2516CF566C66888C6

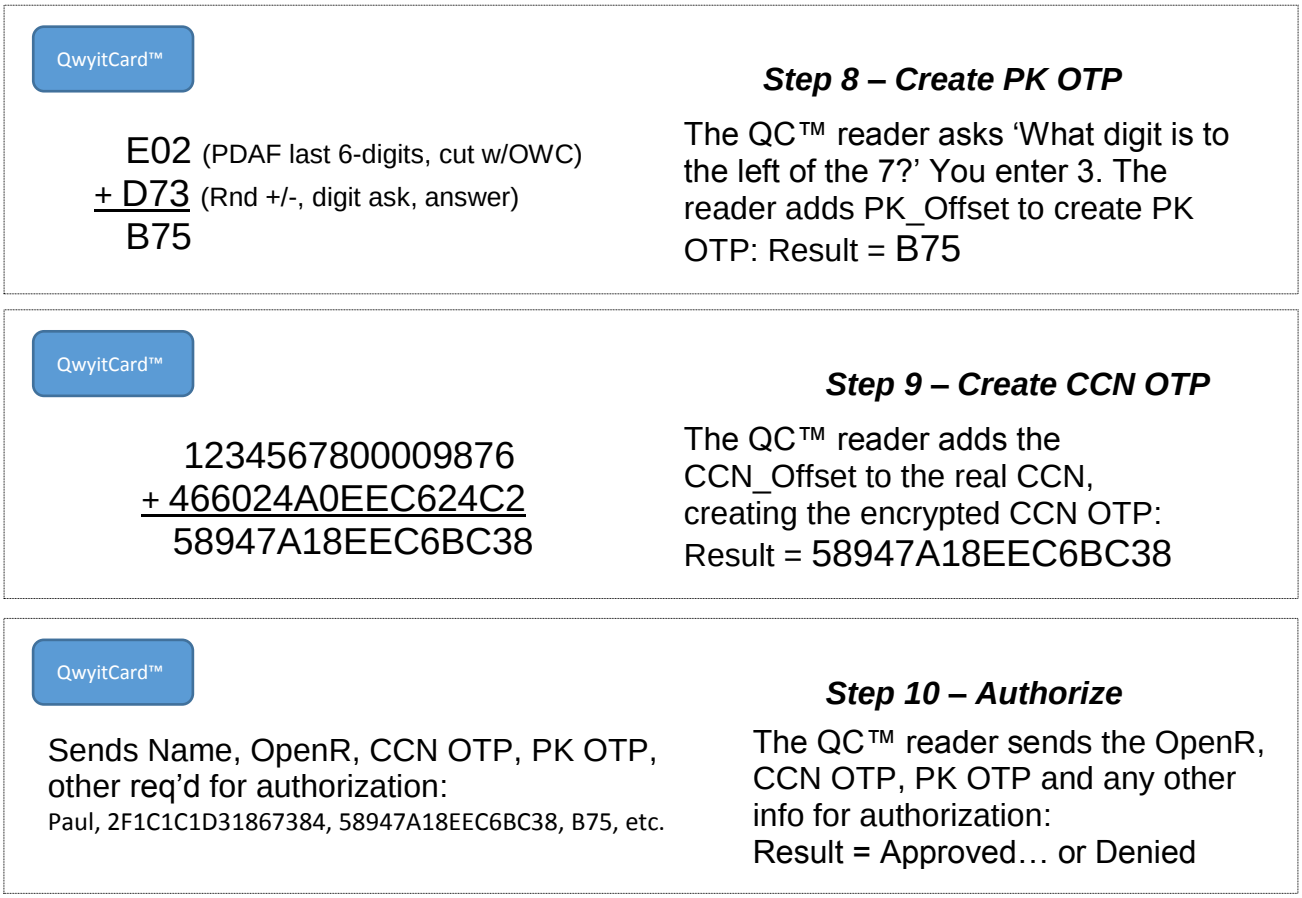
B + 6 = 1, using the D to select the 6
D + 6 = 3, using the C to select the 6
5 + A = F, using the 6 to select the A
2 + 5 = 7, using the E to select the 5 (going back around the value)
6 + B = 1, using the 7 to select the B, etc…

QwyitCard™

13F715798A1364A668A4A2516CF566C66888C6
Add digit pairs:
1 + 3 = 4,
F + 7 = 6,
1 + 5 = 6, etc.

Step 7 – OWC

The QC™ reader performs a OWC to half the PDAF, creating the CCN_Offset (using 1st 32-digits):
Result = 466024A0EEC624C2



Part IV – QwyitCard™ Security

Again, this isn't a mathematics document – but there is enough information, along with the example values, to see the superior potential of using this simple QC™ technique to send a 100% unbreakably secure, unique-every-time, credit card authorization. Yes, the EMV spec covers 'a lot more' than just sending the card for authorization…but most of it can be ignored. Not only because it's boilerplate stuff like error-recovery and attempts to cover all of the various and varying conditions of the reader, the network, etc. – but mostly because it's a derivation of the past, unsuitably layered upon itself so it's almost impossible to tell what is 'required' anymore: What should happen is to start w/the unbreakably unique QC™ technique, get in the lab, and re-build the entire communications authorization process from scratch – keeping in mind the QC™ principles: Fast, Efficient, Simple, Secure. And use Qwyit® unbreakable communications (QwyitTalk™) and unbreakable transaction storage (QwyitStore™).

Let's take a quick look at how to break the QC™ technique:

Scenario	Result
Stolen transactions	**None** – underdetermined equation set: **Unknown/Known** **ECN_d** = MOD16(**ECN**, **PIN**) **OpenRMixed** = MOD16(**OpenR**, **ECN_d**) **PDAF_Result** = PDAF(**ECN_d**, **OpenRMixed**) **CCN_Offset** = OWC(**PDAF_Result**) [1st 16 digits] **PK_Offset** = OWC(**PDAF_Result**) [Last 3 digits] **PK_OTP** = MOD16(**PK Info**, **PK_Offset**) **CCN_OTP** = MOD16(**CCN**, **CCN_Offset**) There is substantial information on these Qwyit® primitives – their operation, their one-way gate protections, the unbreakable security – all can be found at www.qwyit.com
Stolen/lost Card	**None** – PIN strength, 16^4 (65,536 – stronger than decimal) **ECN_d** = MOD16(**ECN**, **PIN**) **OpenRMixed** = MOD16(**OpenR**, **ECN_d**) **PDAF_Result** = PDAF(**ECN_d**, **OpenRMixed**) **CCN_Offset** = OWC(**PDAF_Result**) [1st 16 digits] **PK_Offset** = OWC(**PDAF_Result**) [Last 3 digits] **PK_OTP** = MOD16(**PK Info**, **PK_Offset**) **CCN_OTP** = MOD16(**CCN**, **CCN_Offset**) The 4-digit PIN can easily be expanded to 7-digits, as most people can easily remember 'phone number like' configurations: 3DF-192A. This exponentially expands the PIN strength (16^7 = 268,435,456)
Skimmed/Shimmed/Scammed Card, including shoulder surfed PIN *and* PK single entry – but not in possession of the card, nor transactions	**None** – PK strength remains for any single-digit answer: 10% or better, configuration dependent **ECN_d** = MOD16(**ECN**, **PIN**) **OpenRMixed** = MOD16(**OpenR**, **ECN_d**) **PDAF_Result** = PDAF(**ECN_d**, **OpenRMixed**) **CCN_Offset** = OWC(**PDAF_Result**) [1st 16 digits] **PK_Offset** = OWC(**PDAF_Result**) [Last 3 digits] **PK_OTP** = MOD16(**PK Info**, **PK_Offset**) **CCN_OTP** = MOD16(**CCN**, **CCN_Offset**) The PK info is *only available* on the card – limited probability (10% or less) to answer the challenge correctly, even w/knowledge of how to send it correctly. See Appendix D for complete details.
Stolen/lost Card and Stolen complete transactions	**Crime** – Dictionary attack of all PINs yields **correct PIN**; card possession yields **correct PK** – criminal success. It requires substantial resources, and if QC™ financial transaction system is properly executed, under both QT™ unbreakable communications and QS™ unbreakable storage, as well as partitioned transaction pieces – it can *not* be accomplished by anyone other than an authorized insider – who will be identifiable. *This is the only available break of the QC™ method: and it exists wherever humans participate.* This is the best that can ever be done.

Note that all QC™ transactions are to be sent encrypted from reader to creditor; and Qwyit® most certainly recommends using the QwyitTalk™ unbreakable communications system and any data stored using the QwyitStore™ unbreakable data storage system; therefore stolen transactions simply wouldn't exist. But *even* if the QC™ output was sent openly, without the actual card in hand, any transaction transmissions are 100% unbreakably secured (even if in possession of every single transaction ever created with the same card.)

And even if someone has your card, and even if they have a transaction from that card before they stole it (as card transaction storage at any creditor wouldn't identify the actual card, but rather it should reference a transaction id and account holder, which is then looked up somewhere else for the card number – at least this would be recommended…!), they still need to have the capability to perform dictionary attacks using fake readers, the software to perform it, etc.: this is a sophisticated, targeted attack, where the percentage of fraud committed this way is, arguably, small – since it requires stealing cards from users and transactions from a reader/merchant/Issuer.

The issue of skimming, or any and all methods for attaining 'card data' as the card is being used without actually physically stealing the card in its entirety, even assuming they get all the cards values (CCN and ECN) and PIN, but don't have any stolen transactions nor the Proof Key..OK!...they even get the current PK submission value and know the question that was asked – everything but the PK itself, *still leaves the card protected at the PK guess-percentage that has been deemed acceptable risk by the card Issuer!* If this is, say 10%, because the PK is a single-digit decimal value and they can correctly guess 1 out of 10 tries, the card is still protected with a 1 out of 19 questions (one being known), resulting in 1 out of 10 decimal values – which is still only a 10% fraud rate (or again, whatever level the Issuer has set – 10% being the *highest* risk available in the PK system.) *This is a substantial improvement over existing technology!*

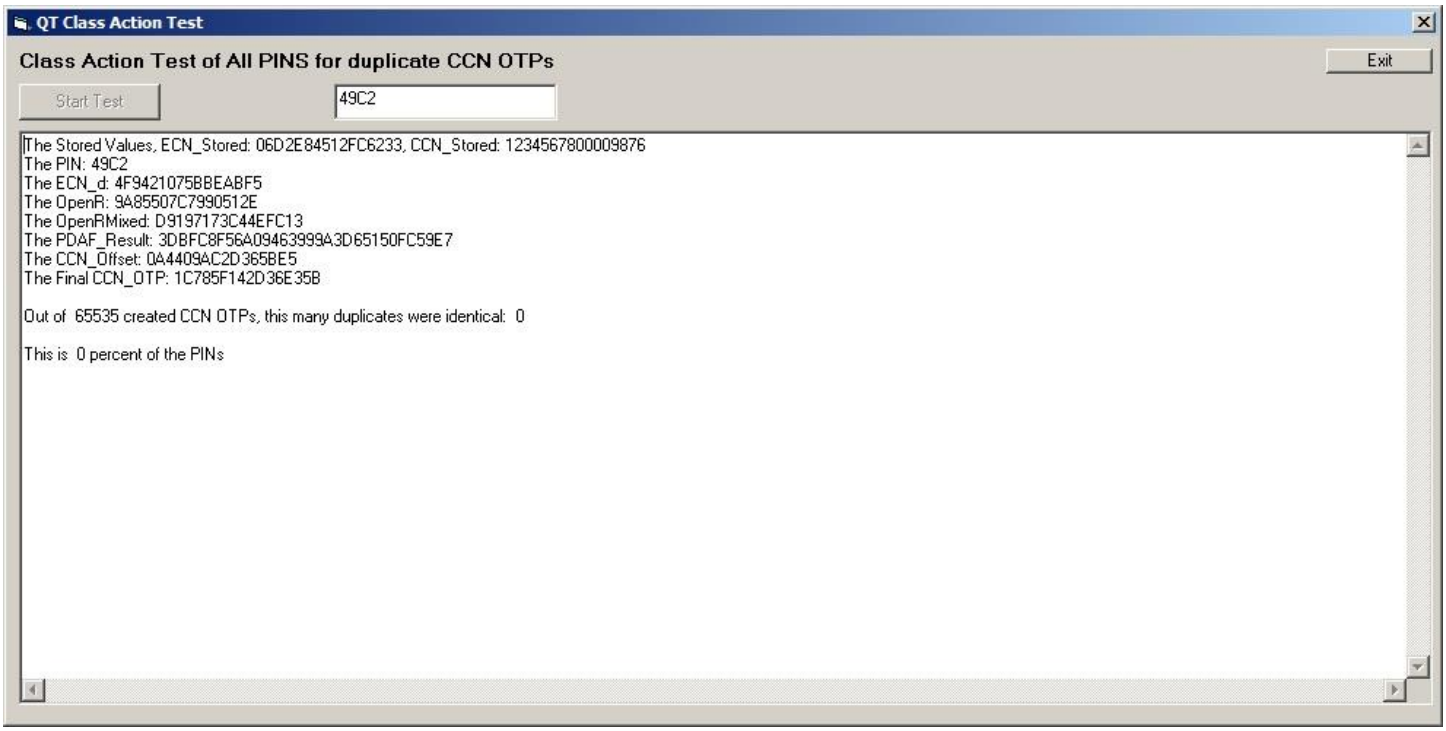
QwyitCard™ Validation

As you see from the above equation sets and example process, the end result is a PIN-protected, OTP version of the CCN (and PK OTP version) sent to the Issuer for authentication and credit approval. The following is a simple validation program operating the QC™ process to test the output for collisions, etc.

PIN Test

Here is a simple program output that uses a static PIN, ECN and CCN to create that CCN OTP, and then tests all of the other possible 4-digit PINS using the same ECN and CCN to see if there are duplicates (the answer is No):

Here is a pic of the test of the PINs:



The QC™ security bottom line is this:

- The QC™ technique delivers an unbreakable CCN version, every time
- Under *every criminal situation up to stealing everything*, there is maximally 100% security, minimally 90%

We even have a faster way to do Steps 6 and 7 – we have primitives called Combine and Extract that cut the processing of a PDAF/OWC in half. The flexibility of the QC™ primitives to incorporate additional requirements and vary the technique is superior.

Part V – Conclusions

The point of this document is to educate, and demonstrate…and possibly to coalesce. If you are a reader that works for a Merchant's Association…or a large Retailer…or maybe even a Creditor looking to expand your business, I hope the message is clear: If reading this has brought about any sense of malaise…any sense that the world of Financial Credit Transactions is being intentionally directed with a heavy, ignorant hand…you're right.

The EMV group has profited from a dire situation of their own creation. They have purposefully imposed a solution to that situation that is not adequate, and already cryptographically broken. They have intentionally limited the credit-reliant economy, and funneled all those customers at point-blank range into their own lap. And then fed them poisoned fruit, while demanding an exorbitant cost for it – and shifted liability for their own ineptitude trapping merchants in a loss any way they proceed.

Time for a change:

- The QC™ technique is an alternative, delivering a superior solution to the exact market-defined problems
- QC™ costs almost nothing, works several orders of magnitude faster, is more flexible in implementation, and is unbreakably secure under every possible circumstance, short of insider fraud and PK knowledge
- How would you like to proceed?

Appendix A – QwyitCard™ Code

The following are the code calls from the reference Qwyit® primitives as outlined in the *Qwyit Reference Guide* to create a unique CCN OTP from the information on a Qwyit-enabled Credit Card, called a QwyitCard™. This includes using the CCN, the ECN, the PK C/R value, and a 4-digit PIN; operating on those, and delivering a CCN_OTP that is unique for every transmission. The Qwyit primitives deliver a mathematic, underdetermined equation set for unbreakably presenting a different CCN every execution.

This unique CCN_OTP provides no information nor process weakness for anyone other than the credit authorizing agent. The provably secure math delivers a 100% authentic token (CCN_OTP) that can only have been created by the owner of the card, who knows the correct PIN. The security of the process is based on the entire key space of the ECN/OpenR (2^{64}), which is substantially larger than the key space of the CCN (10^{16}); the security of the card is based on the strength of the PIN (16^4) and the PK. Both are credible protection.

QC™ Givens

- The QwyitCard™ (QC™) has a 16-decimal-digit CCN (QC™ Credit Card Number)
- The QC™ has an ECN on the 3rd mag stripe, which is a randomly created and assigned 16-digit hex value (64-bits); this is stored using the following QC™ PIN
- The QC™ includes a PIN, which is a randomly created and assigned 4-digit hex value (16-bits) 'stamped' into the stored ECN using a MOD16 of the two values. I.e., QC™ ECN = MOD16[ECN, PIN]
- The QC™ includes a PK, which is an 11-digit value, not included in any magnetic storage

QC™ Process

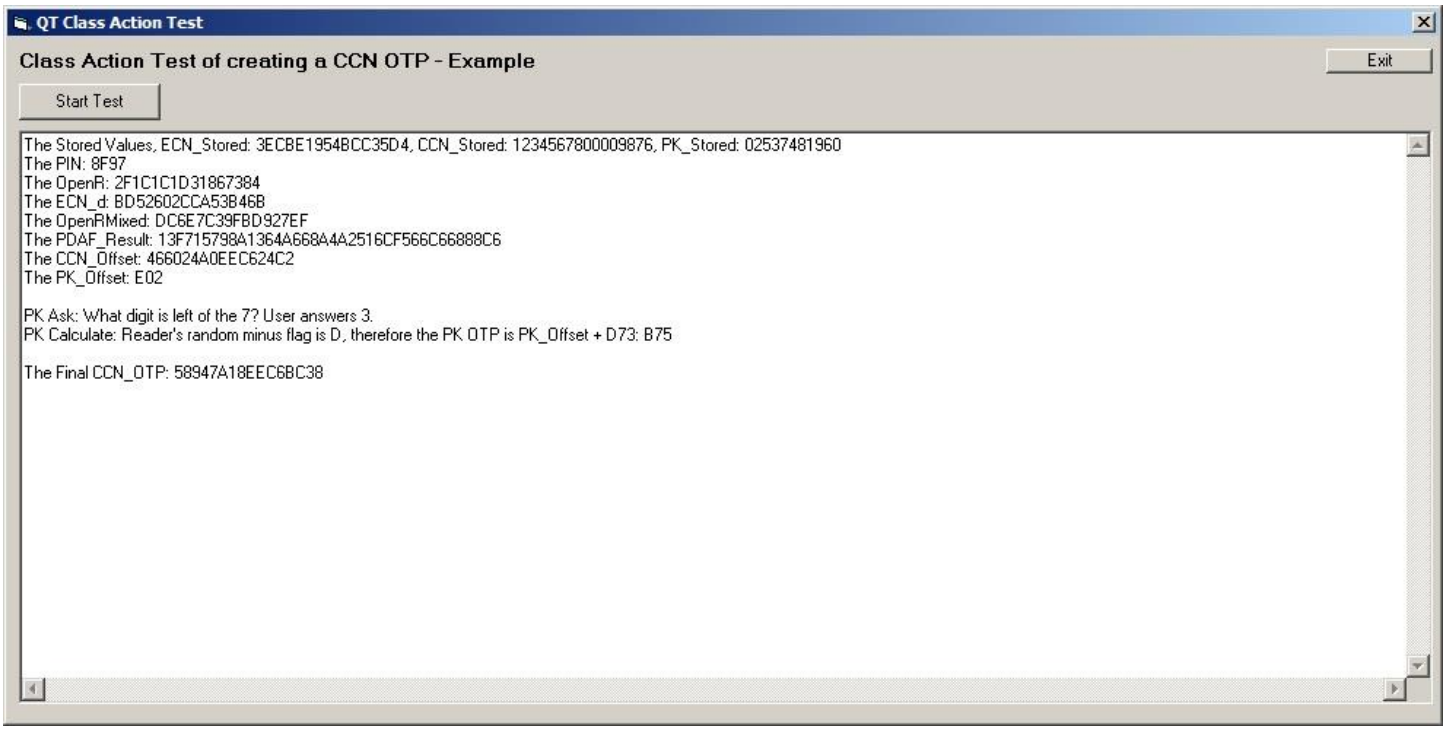
All calls use the reference Qwyit® SDK as outlined in the *Qwyit® Reference Guide*

- QwyitCard™ is inserted into a QC™ reader, reading the CCN_Stored, ECN_Stored values, along w/any other identifying open information (OpenID, etc.) that will be sent with the CCN_OTP to the authorizing agent
- QC™ reader asks user to enter PIN
- A random 16-hex digit OpenR is created
    - sOpenR = GetRandom [16]
- A MOD16D is performed using the entered PIN and the ECN_Stored value
    - ECN_d = MOD16D[ECN_Stored, PIN]
- The OpenR is mixed with the decrypted ECN to create an offset
    - OpenRMixed = MOD16[OpenR, ECN_d]
- A PDAF is performed using the mixed OpenR, and the decrypted ECN
    - PDAF_Result = PDAF[ECN_d, 16, 0, OpenRMixed, 1, 0, 0]
- Since the PDAF is a double-length return, an OWC is performed as a permanent, one-way gate, returning the CCN_Offset and PK_Offset values
    - CCN_Offset = OWC[PDAF_Result, 1] (1st 16 digits), PK_Offset = last 3 digits
- Reader asks for the PK info off the card and calculates the PK OTP using the PK_Offset
    - PK OTP = MOD16[PK_Reply, PK_Offset]
- Lastly, the CCN_OTP is created using a MOD16 of the CCN_Stored and CCN_Offset
    - CCN_OTP = MOD16[CCN_Stored, CCN_Offset]
- The QC™ reader then sends the cardholders OpenID, OpenR, CCN_OTP, and any other required info to the Authorizing Agent for approval/denial

There are 3 equations (a PDAF, an OWC and a MOD16) and only 2 knowns (OpenR, CCN_OTP).

Appendix B – QwyitCard™ Example Output

The following is an example output of the QwyitCard™ process, with explanation:

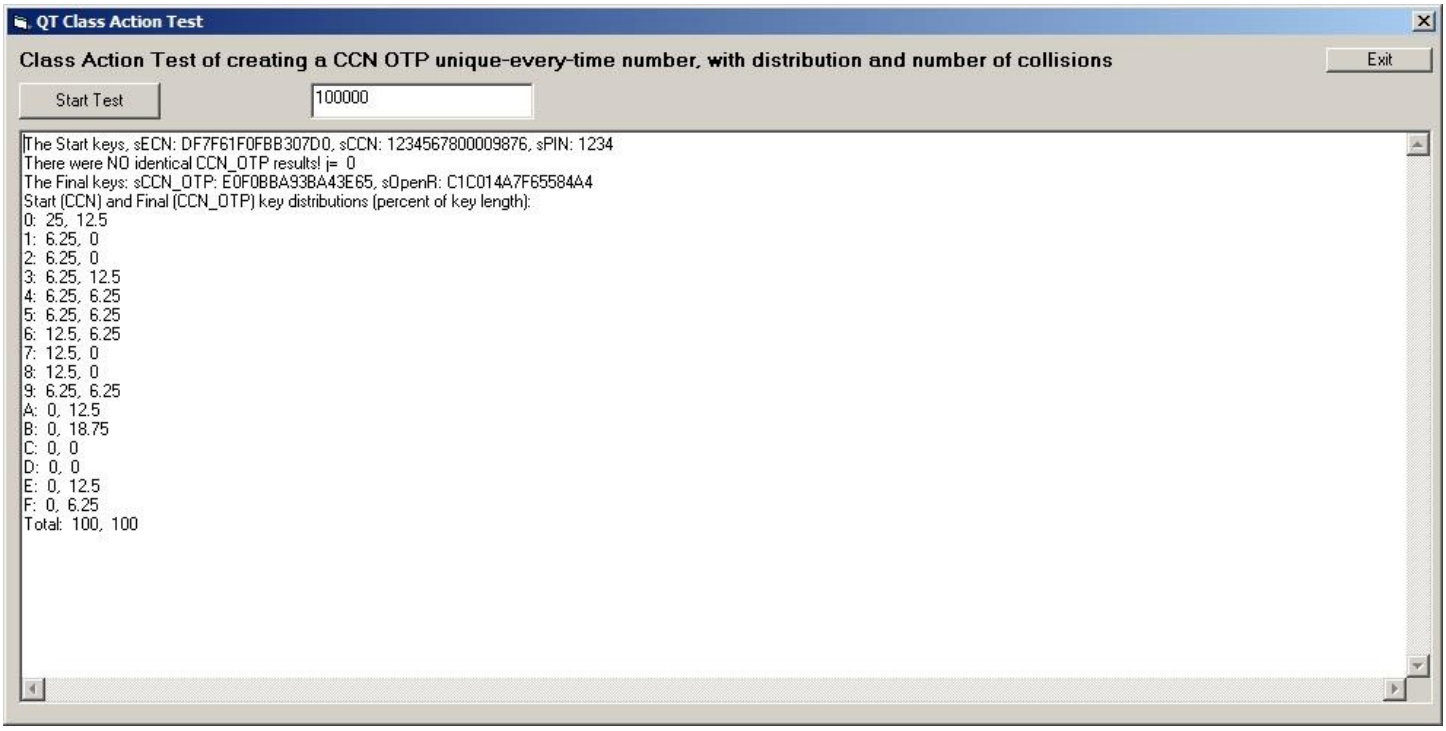


While the exact primitive execution can be found in the Qwyit Reference Guide, here is a brief explanation of each step:

- A random 16-hex digit OpenR is created
  - sOpenR = 2F1C1C1…
- A MOD16D is performed using the entered PIN and the ECN_Stored value
  - ECN_d = MOD16D[3ECBE1…, 8F97] = 3+8 = B, E+F = D, etc. for the full value: BD52602…
- The OpenR is mixed with the decrypted ECN to create an offset
  - OpenRMixed = MOD16[2F1C1C1…, BD52602…] = 2+B = D, F+D = C, etc. for the full value: DC6E7C3…
- A PDAF is performed using the mixed OpenR, and the decrypted ECN
  - PDAF_Result = PDAF[BD52602…, 16, 0, DC6E7C3…, 1, 0, 0] = B+6 = 1, D+6 = 3, etc. for the full value: 13F7157…
- Since the PDAF is a double-length return, an OWC is performed as a permanent, one-way gate, returning the CCN_Offset value
  - CCN_Offset = OWC[13F7157…, 1] = 1+3 = 4, F+7 = 6, etc. for the full value: 466024A…
- Reader asks for the PK info off the card and calculates the PK OTP using the PK_Offset
  - PK OTP = MOD16[E02, D73] = E+D = B, 0+7 = 7, 2+3 = 5 for the full value: B75
- Lastly, the CCN_OTP is created using a MOD16 of the CCN_Stored and CCN_Offset
  - CCN_OTP = MOD16[1234567…, 466024A…] = 1+4 = 5, 2+6 = 8, etc. for the full value: 58947A1…
- The QC™ reader then sends the cardholders OpenID, OpenR, CCN_OTP, and any other required info to the Authorizing Agent for approval/denial
  - TheOpenID, 2F1C1C1…, 58947A1…, B75, other data

Appendix C – CCN OTP Test

The following diagram is the execution output for a program written based on the preceding QC™ process called the Class Action Test program. This program creates any number of CCN OTP outputs (100,000 shown below) and tests for collisions and for Random Rearrangement of the CCN, using unique, new, random OpenR values each time. There are no collisions, and never will be, as long as there are no collisions in the randomly generated OpenR (using the same CCN/ECN values). The digit distribution is shown as per adherence to the Random Rearrangement principles outlined in other Qwyit documents; there is no leaked information in any CCN OTP for use in determining any starting CCN:



Appendix D – The Proof Key Challenge/Response (CVV)

Proof Key Challenge/Response (PK C/R) Technical Description

Method

Requirements:

- QC™ card includes a PK 11-decimal digit number with space between the values on the card back
  - The value is *not* available on the mag stripes
  - The QC™ issuer has knowledge of each Customer's PK
  - PK is a random arrangement of the digits 1-9 non-repeating, bookended by the digit 0 on both ends [There are 9! (362,880) possible arrangements]
    - Example: 0 3 9 8 1 2 5 7 4 6 0

Process:

The Proof Key Challenge/Response (C/R) is between the QC™ reader and the QC™ cardholder. It is performed within the QC™ technique as *Step 8*:

> The PK OTP is created by simple addition of the PK C/R asked/answered info with the last 3-digits of the QC™ technique OWC, *Step 7*. This result is a unique, one-time only value based on the per-transaction generated OpenR and the real PK printed on the card.

- **PK Step 1:** The QC™ reader will randomly select one of the PK Challenge Questions and present it to the QC™ cardholder
  - There are 20 possible questions
    - "What is the value *to the left* of the number *n*?"
      - *n* is any of the PK digits 1-9
      - There are nine (9) available questions of this type
    - "What is the value *to the right* of the number *n*?"
      - *n* is any of the PK digits 1-9
      - There are nine (9) available questions of this type
    - "What is the value *to the left* of the number *0*?"
      - This question asks for the 10th number in the PK string, the value to the *left* of the rightmost (ending) 0
      - There is one (1) available question of this type
    - "What is the value *to the right* of the number *0*?"
      - This question asks for the 1st number in the PK string, the value to the *right* of the leftmost (starting) 0
      - There is one (1) available question of this type
  - The QC™ reader can simply choose either a + (plus, indicating to the *right*) or – (minus, indicating to the *left*), then a value between 0-9. The results will indicate which of the 20 questions to ask
    - For inclusion into the QC™ technique, if a + (plus) is chosen, then a random hex digit between 0-7 will be selected

        - For inclusion into the QC™ technique, if a - (minus) is chosen, then a random hex digit between 8-F will be selected
    - Obviously, a quick process of choosing two random numbers, first a hex value (left or right of), then a decimal value (digit select) would in total pick the question, and prepare for Step 3 below

- **PK Step 2:** The QC™ Cardholder will look-up the correct answer on the QC™ back, and enter the value
    - The possible values are 0-9
    - The QC™ reader may/may not ask if the entered value is correct (user error avoidance prior to submission)

- **PK Step 3:** The QC™ reader will format the PK 3-digit Reply (digit 1 is hex, digits 2 and 3 are decimal)
    - From the question selection and digit creation, the 1st digit is the random +/- selector
    - The 2nd digit is the reference value asked for in the Challenge Question (0-9)
    - The 3rd digit is the QC™ Cardholder's reply (0-9)

- **PK Step 4:** The QC™ reader will perform *Step 8* in the QC™ technique, adding the PK Reply to the PK Offset to return the PK OTP
    - PK_OTP = MOD16[PK_Reply, PK_Offset]

Security

The Proof Key (PK) allows for a 10% guess rate (1 out of the used 0-9 decimal digits), regardless of whether or not the user misunderstands or disregards the challenge question. There are 20 possible questions, and 9! (362,880) possible PK value orders. For every particular card, the answers are definitive; yet knowing 'some' (under any skimmed knowledge circumstances) still leaves enough unknowns to provide security of the card.

The entire set of possibilities for correctly providing an answer for an unknown PK solves the once-known CVV fatal flaw: realizing one PK submission doesn't immediately render the card broken. Under all skimming circumstances, short of 'having the card' (whether in possession or 'holistically' captured), the PK provides a significant improvement in protecting the card from unauthorized use for an extended period of time – as continued wrong submissions of the PK C/R should be flagged by an Issuer, alerting the authorized cardholder.

Should it be desired to lower the guess rate, without elevating the difficulty of the system for the user, simply perform multiple PK C/Rs. Or expand the number of digits to 18, by placing hex digits in between the zeros – and then perform multiple PK C/Rs. Or put multiple PKs on the card – 2 on the front, top and bottom, 2 on the back, top and bottom; then ask for multiple values out of all, some, randomly chosen, varying asks, and all can be handled by extending the entropy of the PDAF/OWC. Or…etc. It's easily possible to lower the guess rate, while maintaining the wide ranging 'key space' of Challenges based on high permutation orderings – a 'perfect' balance can certainly be obtained to level fraudulent skimming to an acceptably low level.

Note: There are any number of 'attacks' that would attempt to 'build' a successful, total PK without having the card; such as the Distributed Guessing Attack. The defeat of any/all of these are straightforward: once discovered, change the method/user involvement/etc. to eliminate it – keeping in mind that this is only a CVV card 'defense'/authorization for CNP transactions. The real defeat of these attacks is to change the way a CNP is performed; and with QC™ primitives/processes, this can easily be accomplished – always using the QC™ principles: Fast, Efficient, Simple, Secure. And use Qwyit® unbreakable communications (QwyitTalk™) and unbreakable transaction storage (QwyitStore™).

Rationale

When I conceptualized the PK C/R 20 years ago for Armor Card, it was based on the exact same rationale as the current CVV/CVV2 numbers that are printed on the card, but not (supposed to be) included in the mag stripe or chip (they sometimes wrongly are, as there are several different versions implemented by all the different Issuers worldwide.) This rationale is how those numbers are used today: for Card Not Present (CNP) types of transactions which don't/can't use the mag stripe or smart chips, such as online and phone purchases (which is why if they are included on the electronic transaction parts of the card (mag stripe and/or chip), they can be stolen during *other* uses of the card, and then used for CNP transactions. This is a fatal flaw implementation!)

These 'Customer Verification Values' (where those CVV initials come from) are intended to connect the card to the user 'overcoming' the 'not present' aspect. Since this is the reason, I am always amazed when I consider the current CVV/CVV2 number complexity: they aren't 'just numbers', they are based on, and are calculated using, a complex series of 'secrets' (like a Primary Account Number (PAN), crypt keys (CVKs), etc.) – as if that somehow gives them 'magical properties' making them 'special': *their purpose is for the user to openly share for authentication – what possible difference does it make <u>how</u> they are calculated?!* Once they are discovered, whether during the *open sharing or sighted*, they are useless – their purpose is in *<u>presentation</u>*, not in *<u>formation</u>*! *Their secret is their physical location during the transaction process, and their transaction process location – they just need to be 'pseudo-random'!* All the calculations by the Issuer to 'prove they are correct' are completely, totally, 100% *<u>nonsense</u>*! When *presented*, are they the numbers *on the card?* That's IT! Actually having something that can be determined *in another way besides having the card (or minimally seeing the numbers) is a security <u>defect</u>, not a security <u>property</u>!*

So what are those properties? In totality, any/every CVV should

- Prove you have the card in your possession (for CNP and/or any transaction deciding to require it)
    - One could just know the values: so making them 'hard to share off of the card' is preferred, if not required (value *complexity*)
    - Make them hard to 'just see and then know' is preferred, if not required (value *substantiveness*)
    - Yet make them easy for the cardholder to obtain (read/see/calculate on/from the card) and present during a transaction (value *simplicity*)
- Provably restrict their access/presentation during any transaction where they are not included
    - Only store them physically on the card, minimally (not electronically, not physically impressionable, etc.)
        - For users with vision hindrances, this is a difficult problem to solve; limiting their usability
    - Limit the risk of simple guessing of the to-be-presented value low enough to be acceptable to the Issuer

You can plainly see that Issuer 'calculations' have nothing to do with CVV value properties, since they're all about what's *on the card*: *complexity on the card*, *substantial presence on the card*, *simplicity on the card* all balanced with the value's *restrictive presence on the card*! Current CVV/CVV2 implementations are woefully lacking, with unnecessarily complex processing. The QC™ technique implementation of the PK C/R establishes a much better balance: simpler in all aspects, while maintaining a long-life even with partial leaked knowledge under card use and processing, with flexibly manageable Issuer risk. Here's how that is delivered:

The above goal of *authenticating you to the card while delivering the best properties balance* requires some kind of Challenge/Response. The simplest way to do this is what the CVV/CVV2 is: print a number on the card, and ask the user to supply it. Don't store it anywhere, and no matter how that number is created, supply it when asked. This 'works': It can *never* be perfect – if it's too difficult for the user, it won't be used; if it's too easy to guess/cheat/steal/subvert/skim (unauthorized obtainment!), it won't be acceptable to the Issuer.

The main problem w/current CVV/CVV2 methods, and what's different about this PK C/R that I proposed years ago, is that the Challenge has no 'key space', no pool of possibilities from which to choose; and the Response is, therefore, of a single static value that has some protection against guessing, but that's it (a 3-digit, 1 in a 1000…4-digit, 1 in 10,000). Current methods fail on complexity and substantiveness; the complexity is in the wrong place (crazily, at the Issuer!) with none on the card itself, as it's only 3 or 4 digits that can easily be seen and remembered, passed along, etc.

The PK C/R process of user-computation (simple and easy to do!) using the Challenge as input to select the Response, turns a static card into a changeable one – the exact goal of the chip cards: they can 'compute'! This ability meets the complexity. The PK C/R large, varyingly-configured static set of values from which only a single (or small) return is selected, delivers a substantive representation on the card. And the simplicity with which the user can easily 'calculate' delivers simple compliance. Lastly, the risk can be set as low as desired, with simple modifications to the process and represented value set; without undue stress for cardholder usability, thus compliance/participation.

In addition, *if* a PK C/R response is logged/stolen and the question is known, there is still 'entropy' in the PK value set to thwart/slow down further/initial unauthorized use – at least long enough to alert the Issuer to multiple-failed attempts at PK C/R submission to suspend the card, and undertake current, acceptable processes and techniques to alert/communicate with the authorized cardholder and remedy the situation. All of these PK C/R included features help deliver a successful, better balance in card authentication when not present. Not all card fraud can be stopped: but using a better CVV technique can certainly substantially reduce it.

PK C/R Epilogue

It should be apparent that the above properties, with slight rewording, are actually *the totality of presenting an authorized user of the credit account being accessed:*

- Prove you are the authorized credit owner in an unbreakable manner to the merchant
- Provide an unbreakable transaction transmission from the merchant to the credit Issuer, including an unbreakable response

If you include the proof, with the amount requested, in an unbreakable transmission to the Issuer, with an unbreakable reply to the Merchant, that's all a credit card is for in the first place!

Aside from this paper detailing a 20-year old, but still applicable, incremental change to the existing half-baked solution using a physical plastic card, Qwyit® now offers unbreakable transmission (QwyitTalk™) and unbreakable storage (QwyitStore™); these two can be combined, along with any/all new methods for user entry of their credit credentials to meet the true goal. We also have an entirely new credit transaction system using those two unbreakable aspects that removes the need for any physical device other than your phone – no readers, nothing – called QwyitCash™.

In total, Qwyit® has everything needed to deliver better credit purchasing: the $*3Billion* spent in the last few years is disgraceful – if not illegally coerced.

Appendix E – Implementation Discussion

Business and Technical Features and Benefits

None of the fraud reduction properties of QwyitCard™ are original (albeit, they were when introduced 22 years ago): a unique transaction number, and some new method providing replication deterrence. In addition to the catastrophic EMV chip cards, there are several smart phone apps that 'do the same thing'; some of which handle payment processing along with presentation. All of these – and the reason that none of them have replaced 'The Credit Card' – suffer from two fatal flaws:

- The solutions are *secondary*; i.e., they require having a *credit card in the first place*
- The solutions don't add any security, even though they claim to; i.e., their methods introduce an underlying pyramid of *new attack points while (supposedly) fixing the original ones*

Instinctively, people understand both of these are just another layer of vulnerability – the Credit Card is still king. [Qwyit® has a methodology to completely replace the *entire credit presentation and processing system, including all the cards and readers* called QwyitCash™. This is an order of magnitude improvement; and since this requires displacement of the entire credit status quo, we've introduced QwyitCard™ as an intermediate step on the road to total secure financial 'messaging'.]

This leads to any new solution proposing the required fraud-reduction properties possessing the following *system properties* in order to surmount proliferation obstacles (unless it is being driven/mandated by the credit Issuers themselves – such as their EMV):

- Whatever infrastructure update is required, it must be simple, zero/low cost, and performed simultaneously with any Card improvements (preferably *before* new cards are put in use)
    - Realistically, this can only be a simple firmware update to existing card readers
        - This means a small-kit SDK code/instructions accompanying suggested 'API' connectivity to the widest possible known/used readers in order to immediately begin processing new cards
- Whatever backend processing is required, it must be simple, low/zero cost, and put in place simultaneously with any Card improvements (preferably *before* new cards are put in use)
    - Same type of SDK/API insertion kit, modeled for existing backend services
    - The insertion must have *minimal impact* on current processes
        - Realistically, this would just be at the 'messaging ends' – the 'send' and 'receive'; then all existing prep (at the reader, which has the above simple compatible SKD/API kit already inserted) and processing (the backend) can proceed normally
- Whatever Card improvements are proposed, credit card users must instinctively know how to participate; i.e., whatever is 'in addition to' simple swiping must immediately be recognizable/understandable

QwyitCard™ meets (*exceeds in italics*) all of these requirements:

1. An improved Credit Card – *NOT a secondary implementation*
    a. Meets all of the fraud-reduction properties

i. Unique transaction data
    1. *Actually mixed into the CCN presentation*
ii. Card is PIN-protected against replication/duplication
    1. *PIN can easily be set at exponential increase over current*
iii. CVV is improved (Proof Key), adjustable to Issuer-required risk
    1. *Actually has lasting protection even after partial discovery*
iv. All properties are based on sound, provable mathematics
    1. *Underdetermined equations – unbreakable/unsolvable over the entire life-cycle use of all cards for all users*

b. Improvements have no Card impacts
    i. Zero cost
    ii. Located on unused, existing card structure
c. Infrastructure processing has no impacts
    i. Less than 100 Milliseconds at reader
        1. *Improved Customer experience – speed, convenience, exactly compatible with existing process*
    ii. Less than 100 Milliseconds at payment processor
        1. *No additional approval time*

2. The SDK S/W for implementing QwyitCard™ is already written in C/C++/Java
    a. Less than 5K in code space
    b. *Firmware push performed easily*
        i. *At reader, at Issuer/processor*
3. Additionally, Qwyit® has tools for improved communications security (QwyitTalk™/QwyitStore™) to upgrade entire approval process transmission security
    a. Unbreakable

The summary benefits of QwyitCard™:

An improved Card, process, implementation, economics, experience
With considered, capable future realizations