Titanium 2.0

New security software from BitArts

Reviewed by Dave Jewell

If you're in the business of creating commercial software, then copy protection is likely to be an issue for you. Whether it's a big or a small issue depends on many factors, such as the price of your product, whether it has mass-market appeal, whether it's a specialised, vertical market, item, etc. As you will doubtless appreciate, there are numerous approaches to the problem of software piracy. A common technique is to disable certain features of a program, restricting complete functionality to those who have paid for it. Another approach is to only work for a certain number of invocations, or for so many days after installation, and so on.

Software protection seems to be becoming an increasingly hot potato: as I write this article, Microsoft has just signed off Windows XP Release Candidate 1, the first Microsoft platform that will incorporate product activation. This means that XP insists on 'calling home' within 14 days of product installation, passing registration key details back to base, and making it problematic for someone else to reinstall the software with that same registration key. Office XP also uses product activation, and even the latest incarnations of dear old Delphi and JBuilder 5 now include mandatory product registration. It's debatable whether these moves are a response to a sudden, huge growth in software piracy, or just a cynical attempt to maximise revenue at the expense of user convenience.

You won't be surprised to hear that, to the best of my knowledge, all four of the above-named products were duly 'cracked' within hours of their release, but that rather misses the point. There are cost/ benefit tradeoffs here: the traditional argument goes that 90% of casual hackers can be defeated using a very simple technique, such as compressing your executable to make decompilation difficult. A much greater effort is needed to deter another 9%, while that final 1% of dedicated anoraks will hack their way through anything you devise and can't, practically speaking, be stopped from doing their worst.

At least, that's the traditional argument. However, new software-only products such as BitArts Titanium 2 are emerging which look set to provide virtually unbreakable levels of copy protection, without requiring the use of hardware 'dongles' and the like. Using the mutation engine contained therein, it becomes impossible to create the sort of generic crack so beloved of hackers. But I'm getting ahead of myself. Let's take a closer look at Titanium 2 and what it claims to deliver.

Titanium 2: Secure Data Delivery

Secure delivery is what it's all about. At its heart, Titanium acts as a wrapper around your program executable. If you're not familiar with the concept of a 'wrapper', suffice to say that Titanium takes your original executable code, encrypts it, compresses it and hides it inside a new EXE file, which then becomes your program from the viewpoint of a naïve end-user. This is a fundamentally similar technique to that used by EXEPACK, Shrinker and other compression tools. However, whereas the emphasis of these tools is on compression, the emphasis of Titanium is on security. Most importantly, only authorised users are able to run the 'wrapped' executable.

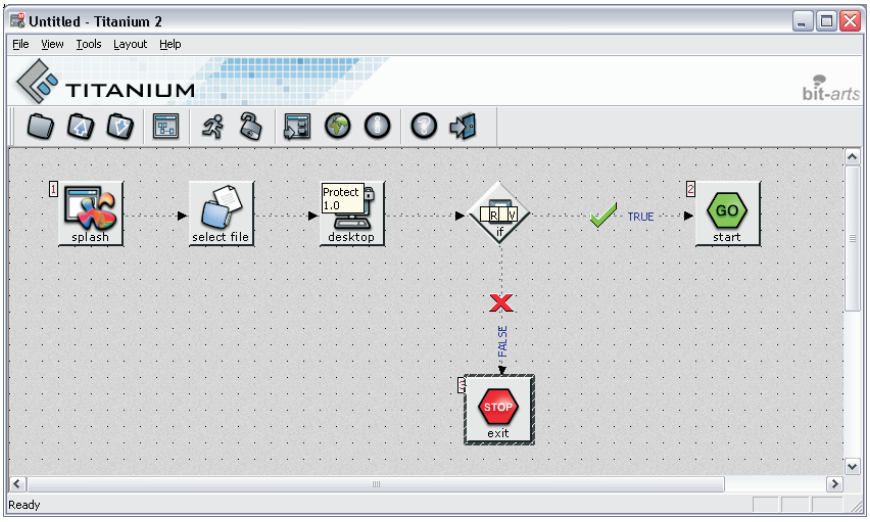
Many shareware and commercial products use some sort of hashing algorithm which takes a username and (optionally) an organisation name, spitting out a special registration key. Only by entering the correct key into a registration dialog is the product unlocked. This is how the ever-popular WINZIP works: just plug in 'Dave Jewell' as the username, and the internal algorithm calculates the correct registration code, refusing to register your program until that code is entered. And therein lies the Achilles heel of this approach: because the hashing algorithm is held locally, it can, albeit with some effort, be found, duplicated and used to produce a 'Keygen'. If you're at all familiar with the 'warez' scene, you'll know that a Keygen is a small, special-purpose program whose only aim in life is to generate registration keys for a specific program.

Titanium cunningly gets around this weakness by removing the generation of unlock codes from the local executable. If our prospective hacker doesn't have access to the algorithm used, then he/she can't duplicate it. This can be done in one of two ways: either manually using a License Manager or automatically using a built-in e-commerce facility. Let's look at the manual situation first.

BitArts supply a License Manager application which can be used to create an unlock code, entering certain information such as a special serial number supplied by the user (he/she sees it in the license dialog when attempting to register the product). This serial number is generated according to certain hardware parameters of the machine so that, if the user attempts to use the same unlock code on a different machine, the unlock attempt will fail. Although the serial number presented to the user is machine-specific, it also exhibits some 'history behaviour'. This is important for the following reason: suppose a user asks for an unlock code to extend the usage of some product for 30 days. After this 30-day period, our devious user might attempt to use the same unlock code to give himself another 30 days of time. What he doesn't realise is that the first time

he used the unlock code, it resulted in a change to the serial number, rendering the unlock code useless for subsequent unlocks.

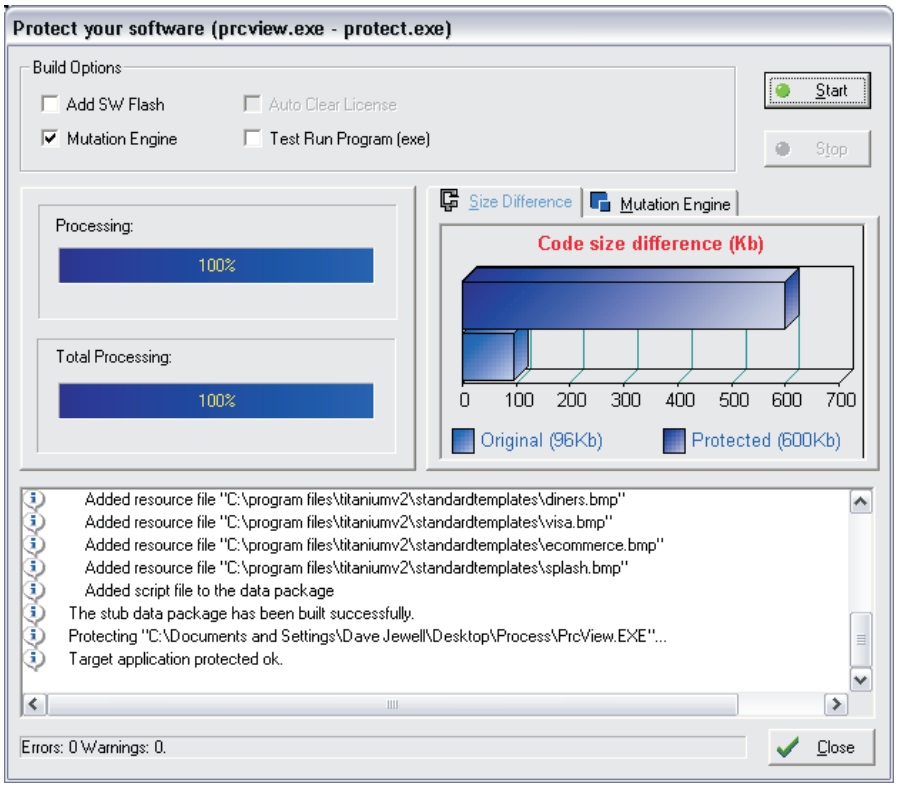
Although the manual approach is great as far as it goes, it does mean that both you and the user have to mess around with serial numbers and unlock codes. However, wouldn't it be great if the whole thing could be automated? For this, BitArts has come up with a new, internet-based authentication scheme which it refers to as Charge Key technology. Using Charge Key, you can embed e-commerce software into the finished executable, allowing the user to register your software online using a credit/debit card. This is done via a secure connection to a Charge Key server which, in turn, communicates with the issuing bank in real time to authorise the transaction. If you go down this route, BitArts allocates you a unique merchant ID which is referenced from within the licensing code. Once a transaction has been authorised, a sales confirmation is automatically emailed to both you and the purchaser, and the software is automatically unlocked.



➤ *Figure 1: Titanium uses an innovative 'Process Flow' system which enables you to visually lay out the desired licensing scheme for your application. Various other desktop elements are included in the package. And yes, it does run under Windows XP!*

This approach adds around 200Kb to the size of the wrapped executable, primarily because of the complexity of the client SSL code which is required for a secure connection to the Charge Key server. You should also be warned that BitArts will charge you 6% of the purchase price of your product for every transaction. However, I think this is a reasonable deal, especially when you bear in mind the 'instant gratification' factor here. Speaking as an e-commerce 'junkie' (at least, where programmer's tools and utilities are concerned) I can testify that you're far more likely to get a sale if it's just a matter of the potential customer pressing a few buttons to auto-register the software online. Waiting for a registration key to be sent via email, or (horrors!) having to pick up the phone and actually *talk* to someone is just too much hassle.

➤ *Figure 2: As you can see, there are build options for adding the Mutation Engine and Flash support to the wrapped executable. Note that when embedding Flash movies, no other runtime support is required: the rendering engine is built in.*



What I've described thus far is the Charge Key Host system, so called because BitArts is effectively giving you (or rather, your customers) access to its Charge Key servers. If you go for this sort of deal, you also receive security certificates which enable you to add, delete and edit product specifications on the BitArts servers. This is done remotely via a web-based management console. You might need this facility if, for example, you change the price of one of your offerings.

In addition to Charge Key Host, there's also the Charge Key Solo system. In this scenario, the Charge Key server software is

installed directly onto one of your own machines running Windows 2000 Server with Microsoft SQL Server. This obviously gives you maximum control over security, and BitArts points out that this approach is ideal for ISVs who are working with resellers, because you've got the option of setting up multiple merchant IDs on the system, one for each reseller.

Go With The Flow!

One of the most innovative and interesting features of Titanium 2 is the Process Flow system which has been designed into the product. On the one hand, the company wanted to provide a high level of flexibility, allowing software developers to create their own custom licensing scenarios. But, on the other hand, it didn't want to develop a full-blown SDK, with all the language-specific hassles and support issues which that entails. What it came up with was the Process Flow system which visually walks you through the licensing process, enabling you to make the thing as simple or as complex as you want.

Let's walk through a simple example to show you how this works. To begin with, you need to launch Titanium itself and click `New` from the `File` menu in order to start a new Titanium project. Project files, incidentally, are saved with the extension `.tdf`. Once you've done this, you'll see a window very much like the one shown in Figure 1. The Process Flow elements can be dragged around the work area, while the connecting arrows follow each element around. There are assorted options for controlling the appearance of the Process Flow area: you can change fonts, text colour, line colour, arrow style, and even the background bitmap. Not hugely relevant to the job in hand, but fun to play with, nevertheless.

In Figure 1, the leftmost item is the splash screen element. This corresponds to the splash screen that the user will see each time he/she runs the application. This isn't mandatory: if you want, you can remove the splash screen altogether, but I'd strongly recommend that you keep it for the simple reason that the Titanium wrapper takes a few seconds to load and verify the program, so you should give the user something to look at during this time. Right-clicking the splash screen element allows you to configure the screen position of the splash, how long it's displayed, and so forth. At this point, I should mention that all the various dialogs displayed by the Titanium wrapper (including the splash screen) are supplied with the product as a number of dialog templates. You can customise these templates for your own use, and of course you can therefore change the splash bitmap. One of the most exciting new features in Titanium 2.0 is the ability to embed Macromedia Flash graphics into your wrapped executable. Using the integrated dialog editor, you can add Flash movies to any dialog of your choice. Variables in the Flash

movie can be mapped onto built-in and custom Titanium variables making it possible to seamlessly integrate the movie into the end-user's experience.

Moving from left to right in Figure 1, next up is the `Select File` item. This enables you to select the executable that's going to be wrapped. At this point in time, Titanium will only protect EXE files: it won't work with DLLs. However, if you really need to protect a DLL, BitArts does have other products that will do the job. Next we come to the not-very-well-named `Desktop` item. This is primarily concerned with the type of protection that you want to apply to your application. You can choose either a fixed or dynamically generated serial number (about which more later) and you can choose from one of the following three protection types:

- Credits: the evaluation software only runs a certain number of times.
- Days: the evaluation software will only run for a certain number of days.
- Date: the evaluation software will stop running after a certain date.

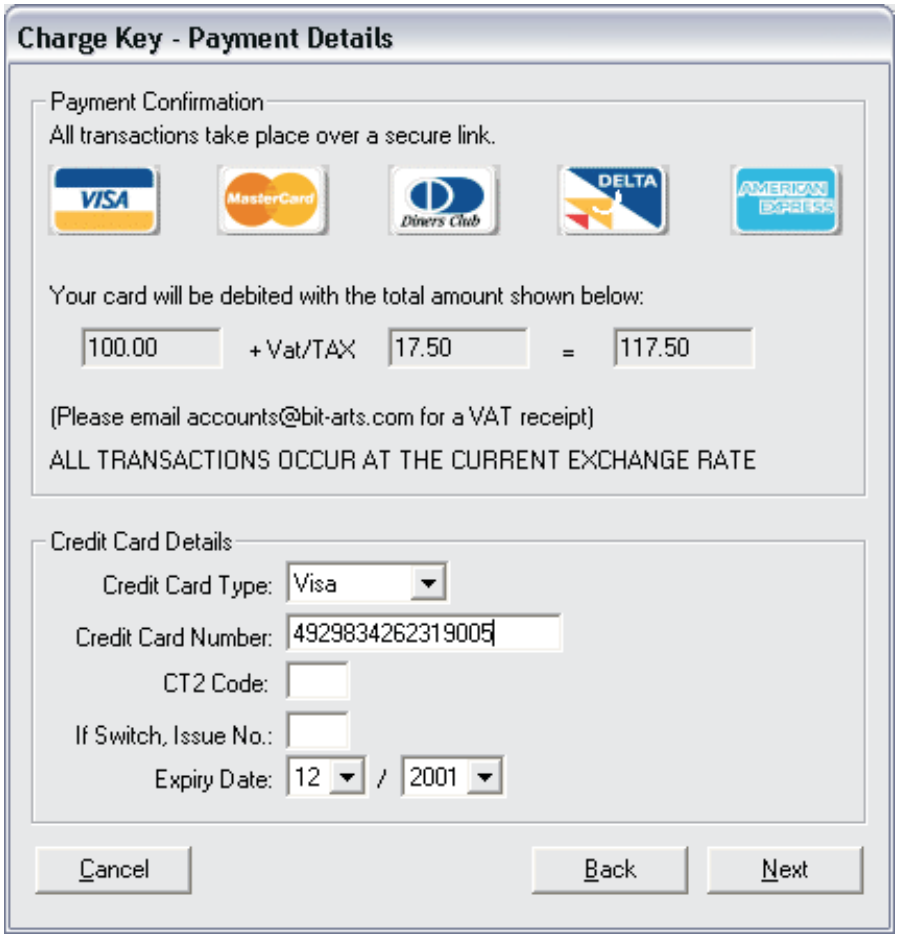
And yes, to answer the obvious question, the security wrapper inside Titanium is smart enough to detect attempts to backdate the system clock in a user's PC, a common ruse to gain more usage.

Typically, you will want to hide the License Details dialog once the user has purchased the software, and there's an option to do this from within the `Desktop` item. You can also choose to redisplay license details once the evaluation license has expired. Most importantly, if you're using the Charge Key system, this dialog provides an entry for you to type in your Merchant ID and Product ID information, (assigned by BitArts) identifying you to the Charge Key server.

As with classical flowchart diagrams, the next, diamond-shaped, box is a `Desktop` conditional item. This always follows the `Desktop` item since it's designed to act on the 'output' of this item. Using the conditional item, you can test if the protection 'state' is `Expired`, `Registered`, `Evaluation`, or `Error`. The meanings of all these should be obvious with the exception of `Error`, which indicates that the program has been tampered with, the PC clock has been backdated, etc. In this case, I set the conditional item to return `True` if the protection state is `Evaluation` or `Registered` and `False` otherwise. As you can see from the diagram, this causes the wrapper to either run the program or terminate respectively.

Much more complex configurations can be created by adding more elements to the Process Flow 'desktop', and there are more available elements than those shown here. For example, you could interpose a Virus Protection element immediately before the wrapped application is to start executing. This performs a rigorous CRC check on the in-memory program image, allowing you to detect and abort unauthorised tampering. The beauty of the Process Flow approach is that such checks can be placed anywhere in your process flow diagram. Another element, Generic Conditional, allows you to set a `True` or `False` condition depending on the state of some internal wrapper variable. For example, you might want to warn the user if he/she has only a few trial credits left. To do this, just test the value of the `%LastDPCreditsLeft%` variable inside your generic conditional and if it's less than, say, 5, redirect the flow of control to a Message Box element which warns the punter that it's nearly time to reach for the plastic.

At this point, astute readers will be thinking that this Process Flow approach has a lot of similarities to the Wise Installer which likewise provides little packages of canned scripting functionality. There are certainly similarities here, but



➤ *Figure 3: This is the default payment details dialog for automated payments via Charge Key. As with all the other Titanium dialogs, you can customise it any way you want. But don't bother jotting down my credit card details, this isn't a valid number!*

overall I prefer the inherently visual nature of the Process Flow approach.

But I digress. Testing things out on a randomly chosen process viewer application (approximately 100Kb before wrapping), the resulting EXE increased in size to around 600Kb. Yes, this is a fairly hefty overhead, but let's make the point that it's a fixed-size overhead which bears no relation to the size of the 'payload'. In other words, with the same set of configuration options, wrapping a 3Mb EXE file will only result in a 3.6Mb EXE and, let's face it, who uses floppy disks to deploy their applications these days? BitArts itself makes the point that Titanium is best suited to full commercial packages and not small shareware utilities, although personally I'd like to make use of Titanium protection regardless of the size of my application!

The End-User's View

OK, so we've now got a wrapped application. Once the user gets past the splash screen, he'll see a license dialog which will indicate the state of the license. A `Continue` button launches the payload application while the `Purchase` button takes you through to the Charge Key system, assuming that this was configured into the wrapper. If you didn't use Charge Key, then the user must be provided with a license key which is entered as a string of hexadecimal digits into the license dialog. Alternatively (less error prone) you can supply it as a small .LIC file which contains the same string of digits. If the license file approach is being used , the security checker automatically detects and validates the file.

From the end-user's point of view, the program behaves just like its 'unwrapped' counterpart, allowing for the slightly longer load time. The original `VERS` resource in the unwrapped EXE file is surfaced so that, for example, examining a wrapped executable's properties from Windows Explorer will show your copyright information and not that of BitArts!

A lot of effort has been put into the defeating of reverse engineering techniques. According to BitArts, the program checks for the presence of in-memory debuggers and the like, performing frequent CRC checks on the internal code to prevent tampering. For the ultimate in security, Titanium incorporates the patent-pending 'Mutation Engine', which is reputed to be the bane of hackers everywhere. The Mutation Engine (not to be confused with the virus-creating technology of the same name!) works by injecting security checking code into random places within the body of your application code.

According to the BitArts blurb, the Mutation Engine is capable of detecting areas of code which are heavily utilised and automatically avoids placing security checks in such code, which would otherwise cause a performance bottleneck. Quite how it manages to do this escapes me: bear in mind that we're talking about static analysis of non-running code, so surely this would require a profiling session before the injection process takes place? But let's not quibble, the important thing is that the software effectively adds numerous security checks within the body of your wrapped code, which alone makes the reconstruction of an unprotected executable into a huge job.

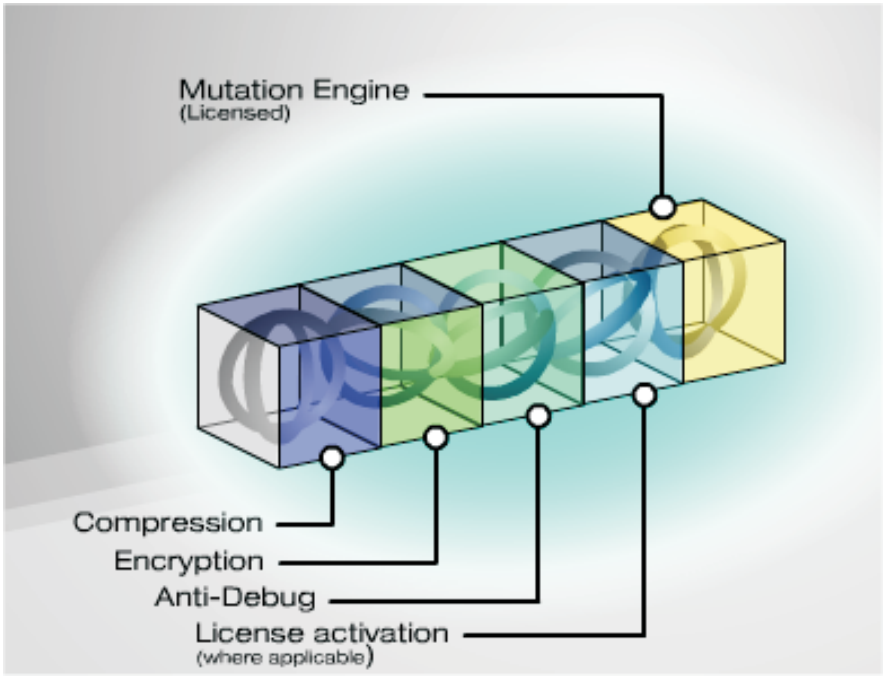
Another Mutation Engine trick is to perform automatic encryption and decryption of security checking code 'on-the-fly', thus making it hard for would-be hackers to track down and identify all the places where such code has been injected. Again, I puzzled over this one, since the frequent references to CRC checking in the BitArts literature would surely mitigate against self-modifying code as well as hacker-modified code? Presumably, the on-the-fly decryption/encryption algorithm used doesn't actually disturb the overall CRC value of a block of encrypted code. Or maybe the security wrapper maintains a list of on-the-fly regions which it specifically

Licensing Options

For the sake of clarity, it's a good idea to summarise the different types of licensing options provided by Titanium. The first decision is whether to have a fixed serial number or a dynamically generated serial number: the latter being created according to certain (unspecified!) characteristics of the PC on which the protected software is running.

Ordinarily, you'll want to use a dynamic serial number, because it's *much* more secure. If you released your software using a fixed serial number, then there's a real possibility that someone will release the unlocking code onto the internet. This will then enable anyone to download the unlock code and unlock all copies of your software: not a nice scenario. Using dynamic serial numbers, different machines will require different unlock codes, but there is one subtle problem here. Suppose you want to place an evaluation version of your product on your website for end-users to download and try out. The problem is that you don't know in advance what will be the resulting dynamic serial number on each user's machine, so you can't supply an evaluation license as part of the download. Does this mean that each punter has got to be emailed an evaluation license before they can even try out the product? Bummer.

Fortunately, BitArts gets around this by providing what they call a Floating Evaluation license. This gives the user a time-limited or credit-limited evaluation license which isn't dependent on the dynamic serial number on his specific PC. However, this serial number is displayed in the license dialog and is used for registration, either manually or via the Charge Key system. This is the ideal configuration when making evaluation software available over the internet. Other options are built into the system, including the ability to limit the number of concurrently running instances of your software in a networked environment.



➤ *Figure 4: This screenshot, taken from the BitArts website, shows the various defences that are incorporated into a Titanium-protected executable, the Mutation Engine being the most powerful.*

ignores during CRC checks? Oh, my brain is starting to hurt.

Conclusions

Well, as with anything else, the proof of the pudding is the important thing. I've trawled the 'warez' newsgroups and other nefarious places, and there's no mention of any sort of generic 'crack' for Titanium and the Mutation Engine. I've seen a wide variety of software packages, some of them very expensive and protected by sophisticated DLLs, but the crack required to remove evaluation limitations is often childishly simple once you find the single point in the code where the evaluation state is tested. I don't believe that Titanium 2 suffers from such glaring weaknesses and I can wholeheartedly recommend it to those who want the ultimate software protection for their products.

For sure, Titanium 2.0 is an expensive package in its own right, but you'll notice that it has been used to protect itself. The significance of this, naturally, is that BitArts must be pretty confident about the level of protection it gives. Needless to say, I haven't found any cracked versions of Titanium lurking on the web. I tried typing 'Titanium crack' into my fave search engine, and all I got was details of someone complaining about fractures in their gold-plated titanium spectacle frames!

Whether I was writing an expensive, specialised, vertical-market application, or a small, mass-market utility like WinZip, I'd want to have my code protected by Titanium 2.0. True, there's that 6% 'tax' on every purchase but, like I said earlier, I reckon this will be more than compensated for by the increased sales resulting from the 'gratification factor' that end-users get from being able to instantly register their chosen product, and all without having to monkey around with hardware dongles. And of course, the increased sales from people who would otherwise simply download a 'warez' version of all your hard work.

I should warn you that the Charge Key Solo option (having your own Charge Key server) is not cheap. BitArts has indicated to me a base price of around $20,000 although the actual figure will depend on your exact requirements, customising the software to meet your precise needs, and so on. The company is currently looking at ways to reduce this figure, possibly through leasing of the server software, and it is also considering ways of bringing down the entry price for small-volume Titanium customers, perhaps through rental, or by applying some sliding scale that's dependent on your own sales volume. If you want more details on the current state of play, you can contact the Charge Key sales team on +44 (0)115-9474568.

Titanium 2.0 costs $999 and is available from BitArts, 18 Friar Lane, Nottingham, NG1 6DQ, or email sales@bit-arts.com. The company website is at www.bit-arts.com (note that it is *not bitarts.com*), where you can get more info on Titanium and the company's other security and compression products.

Dave Jewell is the Technical Editor of *The Delphi Magazine*. You can contact Dave by email at TechEditor@itecuk.com